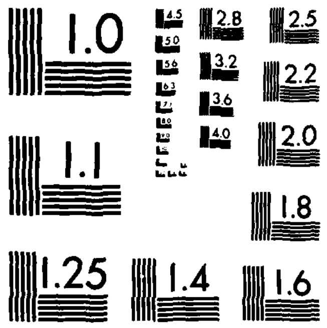
AD-A153 042 DEVELOPMENT OF A CONSOLIDATED DATA BASE FOR STOCK CONTROL AND PRODUCTION..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. A I EL DEIHI 1/2

UNCLASSIFIED DEC 84 AFIT/GCS/MATH/84D-3 F/G 9/2 NL


1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

DEVELOPMENT OF A CONSOLIDATED DATA BASE FOR STOCK CONTROL AND PRODUCTION MANAGEMENT

THESIS

Aly I. El Deihi
Colonel, Egyptian Army

AFIT/GCS/MATH/84D-3

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCS/MATH/84D-3

DEVELOPMENT OF A CONSOLIDATED DATA BASE FOR STOCK CONTROL AND PRODUCTION MANAGEMENT

THESIS

Aly I. El Deihi
Colonel, Egyptian Army

AFIT/GCS/MATH/84D-3

DEVELOPMENT OF A CONSOLIDATED DATA BASE FOR STOCK CONTROL AND PRODUCTION MANAGEMENT

THESIS

Presented to the Faculty of School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science

by

Aly I. El Deihi

Col., Egyptian Army

Graduate Computer Systems

December 1984

Preface

The purpose of this study was to design a data base system for stock control and production management in one of the main logistics depots in Egypt. The use of a centralized data base system has long been recognized by the depot's management as essential for solving the problems associated with the currently used file management system.

This report is limited in scope to design of the conceptual model, and partial implementation of a subset of the data base with the purpose of testing the model. The report includes an application that deals with solving the cutting problem, as one of the major problems in the depot's factories, using data retrieved from the data base.

I would like to express my deep gratitude to my advisor, Dr. Henry Potoczny, for his tremendous moral support, patience, timely guidance, and assistance in times of need. I also wish to thank my committee members, Dr. Albert Moore and Dr. Gary Lamont, for their precise evaluations and recommendations. A word of thanks is also owed to my typist, Mrs. Ladeena Massey, for her effort in typing this thesis. And finally, I wish to thank my wife, Kamilia for her inspiration and her concern on those long times when I was tied to my work.

Aly I. El Deihi

Table of Contents

List of Figures

List of Tables

Abstract

A relational data base for stock control and production management in one of the main logistics depots in Egypt was developed and partially implemented on INGRES.

Top-down analysis of the depot was carried out, entities were determined, and an entity relationship chart was drawn. The areas of stock control and production management were chosen for applying the first data base project.

Information requirements in the two specified areas were analysed, data elements were listed, and the individual user's views were determined. A set of third normal form relations was derived and used as a basis of the design. The conceptual model was developed using a structured technique that yielded the model in a canonical form.

A subset of the conceptual model was mapped to the logical model in relational form, and the logical data base was implemented on INGRES and tested by running some queries against it.

In conjunction with the data base design, linear programming techniques for solving the cutting problem were studied. An efficient algorithm was selected, modified to minimize time and space requirements, and coded in PASCAL. The system was tested by solving some problems, using data retrieved from the data base.

Finally, several recommendations were suggested in order to make the implementation of the data base easier and successful.

I. Introduction

1.1 Background

The main ordnance depot is one of the main logistics depots. The depot consists of a number of warehouses and a number of factories. Warehouses are used for storing thousands of finished items and raw materials, classified into 21 groups. Each group contains items of similar use, and is stored in a separate warehouse. Factories are responsible for fabricating certain types of items. The main tasks of the depot's management are:

1. Purchasing about 80% of the needed items as finished products.
2. Purchasing the raw materials required for fabricating the other 20% of the items, in the related factories.
3. Storing the items, in the condition, and at the levels specified by the logistics headquarters.
4. Supplying the subsidiary depots with items, according to the headquarters policy.
5. Maintaining up to date records about all the mentioned activities.
6. Satisfying the information requirements of the headquarters for periodic reports and ad hoc queries.

In order to deal with these tasks, the departments within the depot maintain and process large amounts of data, that is characterized by continuous growth and change.

In 1979 the depot's management recognized that the rate of growth and change of the data is always increasing. Correspondingly there was a rapid increase in the amount of paperwork and general administrative overhead associated with it.

It was decided that computerizing data storage, management, and processing would solve the problem.

1.2 Previous Efforts

In 1979 the depot's management started a project for data automation. A data processing department was established, and given the task of changing the manual data processing system to a computerized system. The data processing department conducted a study about the manual system, including the different data processing activities, and the information requirements of the headquarters, and the management of each department within the depots. This task was followed by a feasibility study about hardware and software systems available in the market. As a result of this study the following tasks were performed:

1. Installing a PDP-11 computer
2. Design and implementation of a large number of data files and application programs for data storage, update, retrieval and report production. Each department was given the control over its files and programs.

The implementation of the project met with many difficulties, as any project that tends to change a working system to a new one, but the full support of the management and the insistence of the project team led to success. The new system started to work in 1981. In 1982, after implementing the file system for about one year, the management found that the data processing problem was solved partially, but that there were still many problems.

1.3 Problem Statement

The current file system satisfies the requirement of the depot's management to a certain extent. Of course, this system has many advantages over the old manual system. These advantages are:

1. Rapid response to management requests is acquired although not in all cases, and not for all types of reports.
2. High level of accuracy is achieved.
3. Paper work is decreased considerably, and become more manageable.

However, the new system does not solve the problem completely, and possesses some serious disadvantages. These disadvantages are summarized in the following:

1. Considerable redundancy exists in stored data, causing waste in storage space, because each department has its own private files.
2. Too much time is consumed in updating the files. Since one piece of data has to be updated in all files containing it, otherwise the data content of the files will be conflicting, and information supply will be incorrect.
3. Introducing new applications usually entails creating new files, and consequently increasing redundancy more and more, or restructuring some of the existing files and consequently changing application programs.

The problem now is that, it is required to design a data system that avoids the disadvantages of the existing file system. It has been seen that a database could satisfy this requirement.

(2) Candidate Keys

If there is more than one attribute combination possessing the unique identification property, it is called a candidate key. One particular candidate key is selected to be used as a primary key, and the others are called alternate keys.

3. Data Dependency

Designing a data base using the relational model is simply grouping the data elements in a set of relations called the relation scheme. The inputs to the design process are the data elements and the data dependencies, that means which data elements are dependent on which other (1), (5). Different types of dependencies may exist between the data elements, and each has to be taken into consideration during one step of the design process.

(1) Functional Dependency

A set of attributes B of relation R is said to be functionally dependent (FD) on another set A of R if, at any instant of time, each value of A has one and only one value of B associated with it, i.e., given any value of A, the value of B is uniquely determined, and is written as A ---> B.

(2) Full Functional Dependency

A set of attributes B of relation R is said to be fully functionally dependent on another set A of R, if it is functionally dependent on A and not functionally dependent on any proper subset of A.

(3) Transitive Dependency

A set of attributes C of relation R is said to be transitively dependent on another set A of R, if for the three sets of attributes A, B, and C, A ---> B and B ---> C, hence A ---> C, and at the same time B -/-> A.

1. The Tabular Representation

In the relational model, data is represented with the most natural way of representing data to a nonprogramming user, that is the two-dimensional table. Each table is refered to as a relation. Rows of a table are refered to as tuples and columns are refered to as attributes. Tables have the following properties.

(1) Each entry in a table represents one data element.

(2) Columns are homogeneous, i.e., in any column all items are of the same kind, in other words, drawn from the same domain.

(3) Each column is assigned a distinct name.

(4) All rows are distinct.

(5) Both the rows and columns can be viewed in any sequence at any time without affecting either the information content or the semantics of any function using the table.

The major concept from the relational approach used in developing the conceptual model is the normalization process. In order to introduce this process, it is necessary first to present the ideas of keys and data dependency.

2. Keys

(1) Primary Key

The primary key is one or more attributes that has unique values within a relation and thus can be used to identify the tuples of the relation. A primary key that consists of one attribute is called a simple primary key, and that which consists of more than one attribute is called a composite or concatenated primary key. Data elements that are not part of a primary key are referred to as nonprime attributes.

$$\begin{tikzcd}
\text{product number} \arrow[r, bend left, "\text{one}"] & \text{another unique identifier} \arrow[l, bend left, "\text{one}"] \\
\text{product number} \arrow[r, bend left, "\text{one}"] & \text{product name} \arrow[l, bend left, "\text{many}"] \\
\text{product name} \arrow[r, bend left, "\text{many}"] & \text{factory name} \arrow[l, bend left, "\text{many}"]
\end{tikzcd}$$

Fig 2.2. Types of Relationships Among Attributes

2.2.2. The Relational Approach

The relational model describes the data in a different manner than the hierarchical and the network models. These two latter models use pointer-linked data representations. This type of representation can prohibit many changes to data that may be needed as a data base grows, and growth may cause changes in the logical representation of data and hence in the application programs. At the same time, in these models, the logical linkages tend to multiply as new applications are added, and as users request that new forms of queries be answerable with the data. As a result, a high level of complexity will build up in the data base system (10).

The relational model describes the data in a way that (1) can be understood easily by users with no training in programming, (2) makes it possible to add to the data base without changing the existing logical structure, and (3) permits the maximum flexibility in formulating unanticipated or spontaneous inquiries at terminals (12).

The relationships between entities are part of the conceptual model, and they have to be represented in the data base. Any number of entities can participate in a relationship. On the other hand, the same entities can participate in any number of relationships.

2. Relationships Between Attributes

(1) One-to-one

The product number may be a unique identifier for a product. If with the product number, another unique identifier of the product is stored in the same data base, the relationships between the two unique identifiers is one-to-one (Fig 2.2)

(2) One-to-many

The product name and number exist together. There can be many products with the same name, but their numbers will be different. Every product is assigned a unique product number, i.e., to a given product number there corresponds only one name. This relationship is one-to-many (Fig 2.2)

(3) Many-to-many

A number of products with the same name may have been produced in many factories. A number of factories with the same name may have produced many products. The relationship between the attributes product name and factory name is many-to-many (Fig 2.2).

(2) One-to-many

At a given point in time, zero, one or many products are assigned to one production line, but a product is assigned to only one production line. This relationship is denoted by a single-headed arrow going in the "one" direction and a double-headed arrow going in the many direction (Fig 2.1).

(3) Many-to-many

In one day, a factory may have produced several products. On the other hand, a product may have been produced in several factories. This relationship is many-to-many, and is denoted by double-headed arrows (Fig. 2.1).

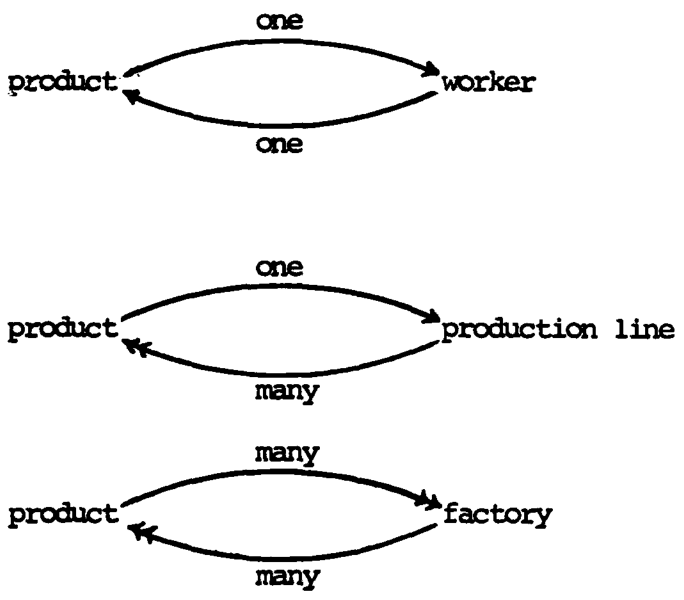


Fig. 2.1. Types of Relationships Among Entities

To develop the conceptual model by this method some concepts from the relational model will be used. However, it yields a data base design that is independent of the implementation approach, that is the conceptual model can be mapped to a relational hierarchical or network data model.

In order to present this general design method, it is necessary at first to introduce two important subjects.

1. The relationships that may exist within a data model, because the main difference among the three types of data models lies in the representation of the relationships between the entities.
2. The concepts of the relational approach, because these concepts constitute the basis of the general design methodology.

2.2.1 <u>Relationships Within a Data Model</u>

A relationship is a mapping or linkage between two sets of data. It can be one-to-one, one-to-many, or many-to-many. Any of these three types of relationships may exist in a data model among entities, among attributes of the same entity, or among attributes of different entities (1).

For example, in a production environment, that consists of several factories, some of the entities can be product, worker, production line, and factory.

1. Relationships Between Entities

   (1) One-to-one

   At a given point of time one product is assigned to one worker. This relationship is one-to-one, and is denoted by a single headed arrow (Fig 2.1).

The entities and data elements are listed in the data dictionary, and the requirements are used in deriving the individual users' views which constitute the first level of the data base system and the input to the process of designing the conceptual model.

2.2 Design of the Conceptual Model

The conceptual model constitutes the second level of the data base system architecture. To develop this model the individual users' views are derived from their requirements and combined in one model. So it is considered as the community user view. It represents the entire information content of the data base, all the entities and the relationships between them, and all the data elements (5).

Frequently the DBA finds himself forced to design the conceptual model as a relational, hierarchical, or network model. This happens in many cases, e.g., a specific DBMS already exits, or the existing CPU can support only one or two DBMS. In this case the DBA has no choice, and he has to design the conceptual model according to the available DBMS.

Ideally, the DBMS should not be a factor in designing the conceptual model. The conceptual model has to be independent of the DBMS, the individual applications, the hardware used for storing the data, and the physical model of the data in the storage media (1). In an enterprise which has not yet purchased a DBMS, the DBA can build a general model. This model has to be mapped after that to a specific data model, that is called the logical model. The mapping process should be done while evaluating different DBMS packages.

For the case of the main depot data base, this second method is the suitable one.

II. Design Approach

The architecture of a data base system that satisfies the needs of multiusers is divided into three general levels: external, conceptual, and internal (5). Consequently designing a data base system is a multi-stage process. It starts after conducting the top-down analysis of the environment and determining the area for applying the data base. In each design stage the data base administrator (DBA) has to perform a specific task. These tasks are:

1. Requirements specification
2. Design of the conceptual model
3. Mapping to the logical model
4. Design of physical model

2.1 Requirements Specification

The purpose of this stage is to determine the information requirements of the management and users, and to unify the naming of the data elements, that will be included in the data base.

All possible means of collecting data may be used. First, the DBA should use a questionnaire to obtain from each level of management a complete list of the data that it needs. Second, all the clerical, operational, and data processing uses of data should be studied. Third, the various types of forms such as bills, reports, and existing data files should be investigated. Interviews are conducted frequently during all these stages, with all levels of management and users. Interviews ensure correctness of the requirements and guarantee users agreement on the naming of entities and data elements.

designing the conceptual model using concepts of the relational model, and mapping of the conceptual model to the logical model is explained. Finally, the design of the physical model is presented briefly.

Chapter III is devoted to the study of the main depot environment. Requirements in the areas of inventory control and production management are specified. The users views are determined and the entities and data items are listed in appendices.

Chapter IV deals with the design of the conceptual model and its mapping to the logical model.

Chapter V deals with the implementation of a subset of the data base, that satisfies the requirements of production planning on INGRES, the DBMS available at AFIT.

Chapter VI contains a presentation of one of the major problems in production planning, the cutting stock problem, and the development of an application program that deals with the solution of this problem. The program is based on operations research techniques, and uses data retrieved from the data base.

Chapter VII contains the conclusion and recommendations for the complete implementation of the data base, and the necessary studies to perform this task.

main depots. This assumption imposes certain requirements to be satisfied.

(1) The development of the data base should be complete, starting from requirement specification to implementation.

(2) The design method should lead to a general model, that can be mapped to any specific data model. This is to give freedom to the designers in selecting the appropriate data model according to the characteristics of the data, and the applications in each main depot.

3. The third assumption is that the major task in this thesis is to design the data base, but w.r.t. implementation it is satisfactory to implement only a subset of it. This assumption is due to the following reasons:

(1) Complete implementation has no benefit far from the application environment.

(2) The task of completely implementing the data base requires time and effort that is beyond the scope of this thesis.

(3) Partial implementation will be included, only to complete the presentation of all the stages of developing the data base.

1.7 Order of Presentation

This thesis is divided into six chapters in addition to the introduction.

Chapter II contains a presentation of the steps of developing a data base system. It starts with describing how to collect information about the data and specify the requirements. A method for

(3) Sharing the Data

Sharing means that existing applications can use the same data. Also it means that new applications can be developed, to operate against the same data, without having to create new files.

(4) Applying Security Restrictions

Having complete control over the data, the data base administrator (DBA) can ensure that access to the data base is through the proper channels, and only for the authorized personnel.

(5) Maintaining Data Integrity

Integrity means ensuring that the data in the data base is accurate. Centralized control helps the DBA to define validation procedures to be carried out whenever any update operation is attempted.

(6) Balancing Between Conflicting Requirements

Knowing the overall requirements, the DBA can structure the data base system to provide an overall service that is best for all users.

(7) Data Independence

Data independence means the immunity of applications to storage structure and access strategy. This means enabling the data base to grow, without the need to change the existing applications.

2. The second assumption is that the work done in this thesis may be used as a guide line for the design of data bases for other

1. The data base system should be easy to use, in order to be understood by the programmers who will constitute the team responsible for the design and implementation of the application programs, and also to be used by the operators without too much difficulty.
2. The design approach should provide high flexibility, to enable step by step implementation, starting with a subset of the data base, and when it is working the data base may be extended one step at a time.

1.6 Assumptions

There are some important assumptions:

1. The first assumption is that the application of data base management techniques offers the optimal solution to the data processing problem of the main depot. This assumption arises from the fact that these techniques provide the main depot with centralized control of its data, leading to many advantages over the current file system. These advantages are:

   (1) Reduction of Redundancy

   Reducing redundancy or at least controlling it in the data base, leads to considerable decrease of waste in storage space.

   (2) Avoiding Inconsistency

   This is a result of the previous point. If redundancy is removed, facts are represented by single entries, and so update operations will not lead to inconsistency.

associated with each entity and their relationships are determined, and represented in a data model.

The three main approaches for data modeling are the hierarchical, network, and relational approaches. For an enterprise with no experience in the data base field, and that has not yet decided the type of data base management system that will be used, it is desirable to build a general model, called the canonical model. This model can be easily mapped to any of the three mentioned models, when the type of the data base management system is decided.

The process of designing the canonical model, and its mapping to the relational model will be dealt with in detail in Chapter II.

1.5 Objectives

The main objective of this thesis is to design a data base for one of the main depots. A data base that contain all the data related to the stored items, and that is suitable for use by all the departments concerned with inventory control and production management.

There are two facts that required other objectives or requirements to be fulfilled in this thesis. The first fact is that the data base designed in this work will be the first data base system applied in the main depots, and may be considered as a guide for building data bases for other depots. The second is that the data processing department of the depot was established recently, hence the programmers have little experience. The major work is done by operators with no programming experience, through interactive user friendly software systems supplied by vendors.

These facts require that the work done in this thesis satisfy the following:

1.4 General Approach

To move to a data base environment, it is essential to have both top-down planning of data, and localized design of data bases. A number of data bases is usually needed because building one data base for a corporation is a complex task. It is far beyond the capability of any one team to design it, and even if it could be designed machine performance considerations would make it unworkable (12).

The main purpose of top-down planning is to make the implemented data systems link together, because incompatible data in separate data bases can prevent the integration of the data needed to generate information needed by management.

In top-down planning the enterprise is studied to develop a model showing its functions and processes, and to determine the entities about which data are stored. An entity relationship chart is drawn. The entities in the chart are then divided into subgroups, each can be implemented in one data base. The subgroup selected for designing the first data base should be one that solves an immediate problem, has fast payoff, and/or is quick and easy to implement.

Ideally top-down planning should take place before the design of individual data bases. But since this task needs a long time to be completed, and there is usually pressure to develop applications, the enterprise can start with a roughly prepared plan, to speed up the implementation of the first data base system. The plan can be refined at each stage before developing new data base systems.

In the design phase the entities of the selected subgroup are subjected to detailed study. Starting from the users' views, the data elements

4. Normalization

The normalization theory is based on the observation that a certain set of relations has better properties in an inserting, updating, and deleting environment than do other sets of relations containing the same data. The normalization process is the discipline of grouping the data into a relation scheme that has the desirable properties (12).

Starting with the user views, which are usually unnormalized data element groups, the designer performs the normalization process on each group step-by-step. Each step yields a set of relations that has better properties than the previous one, and is said to be in a specific normal form (5). The definitions of first, second, and third normal forms are:

(1) First Normal Form

A relation is said to be in the first normal form if and only if it contains only atomic data values.

Relations in the first normal form are obtained by transforming the data elements into a two-dimensional table, and removing the repeated occurrences of them.

(2) Second Normal Form

A relation is said to be in the second normal form if and only if it is in the first normal form and every nonkey attribute is fully dependent on the primary key.

To obtain relations in the second normal form from first normal form relations, an attempt is made to determine the attributes that depend on parts of the total key. If some attributes depend only on part of the key, the key and these attributes are removed into a separate relation.

(3) Third Normal Form

A relation is said to be in the third normal form if and only if it is in the second normal form and every nonkey attribute is nontransitively dependent on the primary key.

Third normal form relations are obtained by separating the attributes, from the second normal form relations, that while dependent only on the key, may have an independent existence in the data base. This is done so that information about these attributes can be entered separately from the relationships in which they are involved.

The first, second, and third normal forms provide successive improvement in the insertion, deletion, and update operations against the data base.

5. The Design Process

The preceding section has introduced the definitions of the normal forms and the steps involved in taking an unnormalized structure and transforming it into the third normal form. To lesign the conceptual model using the normalization process, the following steps have to be performed.

(1) Study the requirements determined in the requirements specification stage (section 2.1), and state the necessary assumptions for it.

The purpose of stating the assumptions is not to solve the environment's problems regarding these assumptions but to reflect them in the conceptual model.

(2) Determine the relationships between the data elements such as identifying the primary key data elements and the nonkey data elements.

(3) Develop the set of third normal form relations that represent the user view of each requirement. Where this is not possible for individual requirement merge data from requirements to establish third normal form relations.

(4) List the distinct third normal form relations derived from all requirements and underline the keys.

It happens frequently that some relations are derived from more than one requirement, but each has to be represented once in the conceptual model.

(5) Draw a conceptual model on the basis of the distinct third normal form relations. In order to keep the graphical representation of the model as simple and clear as possible it is drawn as a structured chart. The relations appear in the chart in levels as follows:

(i) Relations for which the primary key consists of only one data element represent entities. All relations of this type are placed on level 1.

(ii) Relations with a primary key of two data elements represent relationships between two entities. Relations of this type are placed on the second level. If a part of a primary key is not represented as an entity, a new entity relation is generated in level 1.

(iii) The procedure for level 2 is repeated for level 3 on relations with a primary key of three data elements, and so on.

Finally determine the relationships between the relations from the assumptions stated in step 1 and add them to the model.

This process will be used in designing the conceptual model of the main depot data base. All the design steps will be shown in detail in Chapter IV.

2.3 Mapping to the Logical Model

In order to implement the data base, the conceptual model has to be mapped to a logical model based on the relational, hierarchical, or network data models. The mapping to a logical model is required because most of today's DBMS use one of these three models.

2.3.1. Selecting the Model

There are some factors that have to be taken into consideration while deciding the type of the logical model.

1. The Data Relationships

It is known that certain relationships can be represented more efficiently in one model than in others. For example, many-to-many relationships can be represented more efficiently in a network data model than in relational or hierarchical models (1).

2. The Application Environment

The nature of the environment affects to a great extent the choice of a certain model. If the data base field is completely new w.r.t. the environment, it will be better to use the simplest and most flexible model, the relational model. Flexibility will enable implementing the data base step-by-step, and simplicity will enable users without programming background to use the data base.

In this case the designer has to accept some redundancy in the logical representation of the data. But, it is important to stress that this redundancy does not necessarily imply an increase in the storage requirements, because the logical model is concerned with the users views not with the way they are represented in storage.

This is the dominant factor in the case of the main depot data base. Hence, the conceptual model will be mapped to a relational model.

2.3.2. Mapping to the Relational Model

The mapping of the conceptual model onto a relational data model is a relatively easy process. Actually there is no mapping, the same relations appearing in the conceptual model are the same in the logical model. This is due to using the concepts of the relational approach to designing the conceptual model.

2.4 Design of the Physical Model

The physical model is the last stage in the data base design. It is concerned with the way of storing the data on the physical devices. The design is dependent on many factors.

1. The characteristics of the DBMS, such as, the available access methods, the DBMS functions and how the DBMS performs these functions.

2. The characteristics of the direct access devices, because the physical aspects of the data base, such as, the record layout on disk, blocksize, and buffer sizes are associated with these characteristics.

3. The Applications

The number, sizes, and frequency of on-line and batch applications have to be taken into consideration while designing the physical model.

The data base designer tries to optimize the physical model for space and time considerations, taking into account the above factors, and any other factors that affect the performance requirements.

Designing the physical model has to be done after choosing the DBMS. It also needs information about the data base size, which is not available. Hence, it is appropriate to design the physical model in the actual implementation environment.

III. Requirements Analysis

3.1 Study of the Environment

The purpose of studying the environment is to understand the different functions and processes that are performed within the main ordnance depot. This gives the necessary insight to select the area that is most important from the management viewpoint, for applying the first data base project. Understanding the functions and processes will help also in determining the entities that relate to the selected area and that should be represented in the data base.

The main sources of information used in this study are the organizational chart, and the set of manuals that define the procedures followed in performing the different processes within the depot. It is assumed that the information contained in these documents are valid for the following reasons. First, the organizational chart represents the current and actual organization of the depot. It has been developed recently by modifying the old chart, at the time of establishing the new data processing system in 1980. Second, relying on the experience gained from working in different positions in the depot for more than ten years, it can be decided that the manuals describe the actual work done in the main depot. Third, it is not possible to conduct interviews to review these documents, except if the work done in this thesis is performed within the depot.

3.1.1 Environment Description

As have been mentioned in section 1.1, the main depot consists of a number of warehouses and a number of factories, all located in one area. Each factory produces a certain type of items. Warehouses are classified into two groups, one for storing finished items, and one for storing raw materials. Each group of items that have similar use is stored in one warehouse. The warehouse internally is divided into a number of sections, e.g., the clothes warehouse is divided into the woolen clothes section, the cotton clothes section, and so. The major tasks of the depot's management are acquisition, fabrication, storage, and supply.

3.1.2 Functions and Processes

In order to deal with its tasks the depot's management performs different functions. These functions and their associated processes are:

1. Purchase

The major way for items acquisition and replenishment of warehouses is the purchase from vendors, who supply the depot with both finished items and raw materials. The purchase cycle is as follows:

(1) A call for bids about certain items is advertised.

(2) Vendors respond by offering their bids.

(3) Bids are studied, and contracts are made with the vendors whose bids are accepted.

(4) Vendors deliver the items, according to the contracts conditions, to a transiant warehouse, to be stored temporarily for inspection.

(5) Items are inspected, and according to the inspection results, they are delivered to the permanent warehouse or returned to the vendors

2. Production

The second source of warehouse replinishment is the items produced in the related factories.

(1) A factory receives a production order from the depot's management, defining the items and quantities to be produced.

(2) The factory withdraws the raw materials from the raw materials warehouses, and fabricates the required items.

(3) The produced items are delivered to the finished items warehouses.

3. Inspection

The inspection department maintains catalogs containing the standard specifications of all items in the depot.

(1) Items delivered to the depot by vendors are kept in a temporary warehouse, until they are inspected, and their specifications are compared to standards.

(2) Items which satisfy the standard specifications are transferred to the permanent warehouses, and those which do not satisfy the specifications are returned to the vendors.

4. Storage

Items are stored in the warehouses until they are delivered to the subsidiary depots according to the headquarters' orders.

(1) For each item there is a quantity called the critical storage level. This quantity represents the least amount of the item that must exist in the warehouse.

(2) When the quantity of an item reaches this level, the headquarters stops withdrawing from that item, until the warehouse is replinished with a new quantity of the item.

5. Supply

The depot's management receives orders from the headquarters daily to supply the subsidiary depots.

(1) The orders are delivered to the appropriate warehouses.

(2) Each warehouse prepares the required quantities for shipment, and informs the depot's management to assign the transportation means.

(3) The items are shipped to the subsidiary depots.

6. Specifications

The specifications department is responsible for deciding the standard specifications of the items.

(1) When the headquarters decides to introduce a new item, the specifications department defines the item specifications according to its use conditions.

(2) The item is assigned to the appropriate warehouse, and is given an item number after the last item in the items' list of that warehouse.

7. Sales

Sometimes, it is decided to cancel an item; in this case the existing quantity of that item is transferred to a transient warehouse to be sold in the local market, and the item is deleted from the items' list of the concerned warehouse.

3.1.3 The First Date Base Project

It is clear that all the functions, performed within the depot, are concerned mainly with the items. Thus, items represent the most important entity, and its data has to be the core of the first data base project. Correct and timely information about items is necessary for management to deal with the functions and processes in two areas, inventory control and production management. Hence, the data base should satisfy the information requirements of the depot's management in these two areas. It will be called the inventory control and production management data base.

3.1.4 Entities

The inventory control and production management data base has to contain the data about the basic entity, items, and also about the other entities that have relationships with it. Studying the functions and processes makes it easy to determine the entities that relate to the specified areas and their relationships. These entities and the relationships among them are shown in the entity relationship chart, figure 3.1.

There are some other entities such as personnel and transportation means, but those have no effect on dealing with the tasks of inventory control or production management. Hence they are not included in this data base to keep it somewhat simple. These entities and others may be added to the data base later after this first data base project is complete.

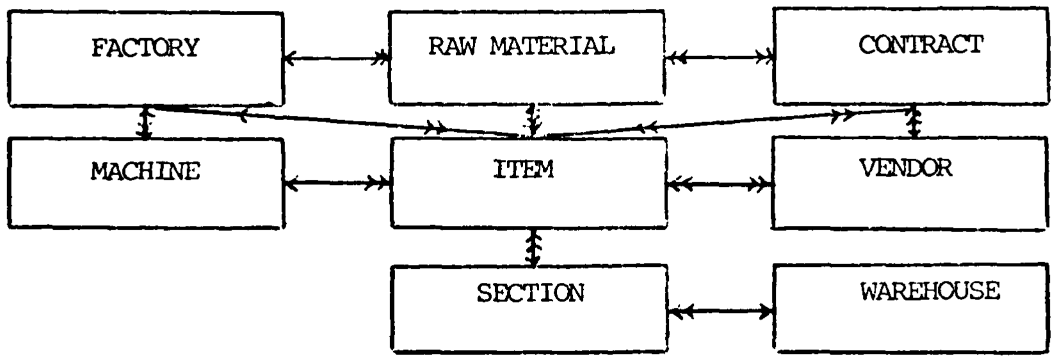


Fig. 3.1. Entity Relationship chart

The description of these entities and the relationships among them are given in Appendix A.

3.2 Requirements Specification

The purpose of the requirements specification is to gather as much information as possible about the data associated with the specified entities, and to analyse this data to achieve the following:

1. Determining the information requirements of the management, in the specific areas of inventory control and production management.
2. Understanding the semantics of the data and stating the correct assumptions about it.
3. Obtaining a list of the data elements that will be included in the data base, to be put in the data dictionary.

The assumptions about data will be stated to help in determining the relationships between the data elements. The requirements and the data relationships will constitute the inputs to the process of designing the data base system.

The main source of gathering information about data are the forms and reports used in the areas of inventory control and production management. Another source is the management queries, which are determined relying on the experience gained from working in the depot. For the same reason stated in the environment study, of being unable to conduct interviews, it is assumed that the information contained in these sources is valid. The major concern here was to extract and study the essential set of documents that represent all the requirements of data in the specific areas. This will make the data base as complete as possible. Any other requirements that may be missed and not taken into consideration will be secondary requirements, and can be generated from the same data base, or at most by addition of a small number of relations to the data base.

3.2.1 Available Information

The forms, reports, and queries were studied. The names of the data element repeated with different names in different places were unified. The unified names were chosen randomly, but this point was not stressed, since these names may be changed when conducting some interviews before implementing the data base in the home environment.

The set of forms, reports, and queries that represent the major requirements are given in Appendix B . The data elements are listed in Appendix C. The following is the description of the forms and reports given in Appendix B.

1. Situation of Vendor's Contracts

   This report is printed for the management about all the contracts of one vendor, at its request. It contains all the information registered in the contracts about the items, quantities, dates of delivery, and the monetary situation.

2. Situation of Item's Contracts

   This report is printed for the management as the previous one, but about all the contracts of one item. It contains all the information about the contracts' numbers, vendors, quantities delivered and quantities under delivery, and delivery dates.

3. Vendors List of Item

   This report is printed for the management at its request. It contains all the information about the vendors of an item whether they have contracts at the time of the report or not. For each vendor, the report contains the vendor number, name, address, telephone number, and the vendor's capacity of the item which is the maximum quantity he can contract for it.

4. Weekly Machine Situation

   This report is printed for the management weekly about the status of the machines in each factory. It contains for each machine the machine number, name, status which is either working or idle, and the stopping date for the idle machines.

5. Production Data of Item

   This report is printed for the management to be sent with the production order to the concerned factory. It contains information required for preparing the production plan, the essential raw material, the item's sizes, the required material for each size, and the percentage of each size in the whole production.

6. Production Order

   Production order is printed for the management to be sent as the previous report to the factory. It contains information required for preparing the production plan. It is used also for withdrawing the required quantity of raw material from the warehouses.

7. Total Weekly Transactions

   A total weekly transactions report is printed for each section within the warehouses. The report is a summary of the transactions that occured during the week. Each item has one line in the report. In addition to the item

4.4 <u>The Logical Model</u>

The conceptual model in Appendix F can be mapped to any one of the three main data models, the relational, the hierarchical, or the network. But, as have been mentioned in section 2.3, the nature of the environment requires the use of the relational model.

Mapping the conceptual model to a relational data model is a relatively easy process. Every box from the conceptual model becomes a relation or a table, and the user is supplied with the user view which is in a table format. This is due to using a relational approach in designing the conceptual model.

For example the boxes 3, 4, 5, 10, 11, and 12 that constitute a subset of the conceptual model of Appendix F were transformed to six relations in table format, as shown in Appendix G. These six relations represent a subset of the logical model. The complete logical model consists of the relations obtained from all the boxes in the conceptual model.

The above six relations were selected deliberately, because they represent the subset that satisfies the requirements in one area, the production management. It would be suitable to start the implementation of the data base with this subset. Later, when this part of the data base works, the other relations in the logical model can be added, to satisfy the requirements in the other area, the inventory control area.

the relationship between the two entities CONTRACT and ITEM. Relation 8 was connected to the two relations CONTRACT and ITEM. The single-headed and double-headed arrows between these relations mean that a given contract may contain many items and a given item may exist in many contracts.

In the same way the connections between level 2 and level 1 relations were established. But, the DATE, MACHINE, and SIZE participate in the third normal form relations of level 2 as part of the primary keys and are not represented as entities anywhere. To show the participation of DATE, MACHINE, and SIZE in the primary keys and to establish the relationships separate relations were created for them at level 1.

3. Only relation 14 has a primary key of three data elements. This relation was placed on level 3, and connected to the appropriate relations in level 1 by the same procedure performed for level 2.

4. The arrows drawn on the top of level 1 represent the relationships between SECTION and WAREHOUSE, between ITEM and SECTION, and between ITEM and FACTORY that is a warehouse has many sections but a section belongs to a specific warehouse, a section contains many items but a given item exists only in one section, and a factory produces many items but a given item is produced in only one factory.

These arrows on the top of level 1 were introduced after the conceptual model for the total set of third normal form relations has been drawn. The arrows represent the relationships that are fundamental to the main depot and are not evident in the total set of third normal form relations based on the reports under consideration.

7. Relations 24, 28, 32, 35, 38, 41 and 45 are identicals

SEC-NO <<---> SH-NAME,WH-NAME

WH-NAME does not exist as a data element in these relations, but it is included as a nonkey attribute for establishing the connection between the warehouse and its sections.

8. Realtions 4 and 9 are identical:

CONT-NO-ITEM-NO <<---> CONT-QTY,DELQTY,QTYUDEL

The remaining relations in Appendix D, 13, 15, 18, 22, 43, and 26 are distinct and they can be carried over. The relations resulting from the merging process and the distinct relations are listed in Appendix E. The distinct relations were given the numbers 9, 10, 11, 12, 13, and 14. The first seven relations represent entities and the latter seven represent relationships between entities.

4.3 Drawing the Conceptual Model

Relations 1 to 14 listed in Appendix E are represented in a pictorial format in Appendix F as follows:

1. Relations for which the primary key consists of only one data element represent entities. Relations 1, 2, 3, 4, 5, 6, and 7 represent the entities VENDOR,CONTRACT,ITEM,FACTORY,RAW MATERIAL,WAREHOUSE, and SECTION, respectively. All relations of this type were placed on level 1, each represented by a box having the same number of relation. The data elements were written inside the boxes, and the primary keys were underlined.

2. Relations 8, 9, 10, 11, 12, and 13 with primary keys of two data elements were placed on level 2. The compound keys of these relations represent relationships between two entities. For example the compound key of relation 8 (ITEM CONTRACTS) is CONT-NO,ITEM-NO. This key represents

1. Relations 1 and 12 are identical and relation 8 has the same key as them. Combining these three relations resulted in:

<u>VEND-NO</u> <---> VEND-NAME,ADDRESS,PHONE-NO

2. Relations 2 and 7 have identical keys. The combination of these two relations resulted in:

<u>CONT-NO</u> <<---> VEND-NO, TOTAL , PAID , REST

3. Relations 3, 11, and 16 are identical, and relations 25 and 46 are identical and the two sets have identical keys. Relations 6, 20, 29, 33, 36, 39, 40, and 42 are not identical to any of the first two sets, however, their keys are identical to the keys of the relations of these sets. Combining all these relations resulted in:

<u>ITEM-NO</u> <<---> ITEM-NAME,ITEM-UNIT,UNIT-PRICE,WH-CAP,CRL-QTY,
QTY-ON-HAND,T-QTYUDEL,DUR-TIME,DESC,SPECS,FACT-NAME,
DLW

4. Relations 14 and 19 have identical keys. Combining these two relations resulted in:

<u>FACT-NAME</u> <--> FACT-MGR,BUILD-NO

5. Relations 17 and 21 have identical keys. The combination of these two relations resulted in:

<u>RM-NO</u> <<---> RM-NAME,RUNIT-PRICE

6. Relations 23, 27, 31, 34, 37 and 44 are all identical, and they can be represented in the conceptual model by any of them as:

<u>WH-NAME</u> <---> WH-MGR-,BUILD-NO

(2') <u>CONT-NO</u> <<---> VEND-NO

(2") <u>VEND-NO</u> <<---> VEND-NAME

The five third normal form relations for the end user's view from the second report "Situation of Item's contracts" are:

(1) <u>ITEM-NO</u> <<---> ITEM-NAME,T-QTYUDEL

(2) <u>CONT-NO</u> <<---> VEND-NO

(3) <u>VEND-NO</u> <<---> VEND-NAME

(4) <u>CONT-NO,ITEM-NO</u> <<---> CONT-QTY,DEL-QTY,QTYUDEL

(5) <u>CONT-NO,ITEM-NO,DEL-DATE</u> <<---> QTYTBDEL

All the other reports given in Appendix B were treated similarly. For each report the relationships between its data elements were determined and the third normal form relations representing the user's view were developed. The relations were examined for satisfying third normality several times, until a satisfactory confidence was gained that all were in the third normal form. The users' views are listed in Appendix D.

4.2 Third Normal Form Relations

The third normal form relations that represent the individual user's views, and that are listed in Appendix D are 46 relations. Some of these relations are distinct, but some other are either repeated in different relations or have the same primary key. The relations were merged to obtain the set of third normal form relations that represent the conceptual view. The merging process was performed as described below and the obtained third normal form relations were listed in Appendix E.

2. Situation of Item's Contracts

The data elements representing the entities of this report are:

ITEM-NO,ITEM-NAME,DATE,CONT-NO,

VEND-NO,VEND-NAME,CONT-QTY,DEL-QTY

QTYUDEL,QTYTBDEL,DEL-DATE,T-QTUDEL

The relationships between these data elements are:

(1) <u>ITEM-NO</u> <<---> ITEM-NAME,T-QTYUDEL

For a given ITEM-NO there is only one ITEM-NAME and one total quantity under delivery (T-QTYUDEL). But for a given item name there may be many item numbers.

(2) <u>CONT-NO</u> <<---> VEND-NO,VEND-NAME

For a given CONT-NO there is only one vendor with one VEND-NO and one VEND-NAME. But there may be many contracts for the same vendor

(3) <u>CONT-NO,ITEM-NO</u> <<---> CONT-QTY,DEL-QTY,QTYUDEL

This is the same relation (4) in the previous report "Situation of Vendor's Contracts".

(4) <u>CONT-NO,ITEM-NO,DEL-DATE</u> <<---> QTYTBDEL

This the same relation (5') in the previous report.

Relations 1, 3, and 4 are in the third normal form. Relation 2 is only in the second normal form, because the nonkey data elements are fully functionally dependent on the primary key, but there is a hidden transitive dependency between the two nonkey data elements. VEND-NAME is dependent on the VEND-NO.

The solution to this problem is to remove the transitively dependent data element VEND-NAME to a separate relation with a primary key VEND-NO. Relation 2 is thus replaced by the two third normal form relations.

Relations 1, 2, 3 and 4 are in the third normal form, because the nonkey data elements from these relations require the full keys for their identification, and there is no transitive dependency between them. But relation 5 is not even in the first normal form, because the mapping is many-to-many. The first normal form requires the mapping between the primary key and the nonkey data elements to be one-to-one or one-to-many. The second normal form requires that the nonkey data elements have the full primary key for their unique identification. And the third normal form requires that there be no transitive dependency between the nonkey data elements.

Relation 5 can be transfered into a third normal form relation if the primary key is further qualified, that is if the primary key is further compounded with DEL-DATE as follows:

(5') <u>CONT-NO, ITEM-NO, DEL-DATE</u> <<---> QTYTBDEL

Relation 5' is now in the third normal form. There is one quantity to be delivered from a given item in a given contract at a given date.

The five third normal form relations for the end user's view from the first report "situation of vendor's contracts" are:

(1) <u>VEND-NO</u> <---> VEND-NAME, ADDRESS, PHONE-NO

(2) <u>CONT-NO</u> <<---> TOTAL , PAID , REST

(3) <u>ITEM-NO</u> <<---> ITEM-NAME

(4) <u>CONT-NO,ITEM NO</u> <<---> CONT-QTY,DEL-QTY,QTYUDEL

(5) <u>CONT-NO,ITEM-NO,DEL-DATE</u> <<---> QTYTBDEL

(1) VEND-NO <——> VEND-NAME, ADDRESS, PHONE-NO

For a given VEND-NO there is only one VEND-NAME, one address, and one PHONE-NO and vice versa. This is a one-to-one mapping represented by a one headed arrow in both directions. All data elements in this relation are candidate keys. VEND-NO is picked to be the primary key.

(2) CONT-NO <<——> TOTAL , PAID , REST

For a given CONT-NO there is only one TOTAL , PAID , REST . But there may be many contracts with the same TOTAL , PAID , REST . This is a one-to-many mapping, represented by a single headed arrow in one direction and a doubled-headed arrow in the other direction.

(3) ITEM-NO <<——> ITEM-NAME

For a given ITEM-NO there is only one ITEM-NAME. But there may be many items with the same name, and each has one ITEM-NO.

(4) CONT-NO, ITEM-NO <<——> CONT-QTY, DEL-QTY, QTYUDEL

In a given contract and for a given item there is only one quantity contracted for (CONT-QTY) and one delivered quantity (DEL-QTY) which is equal to the sum of all quantities delivered until the date of the report. But there may be one item having the same CONT-QTY or DEL-QTY or both, in many contracts.

(5) CONT-NO, ITEM-NO <<——>> DEL-DATE, QTYTBDEL

This relation is separated from relation (4) although it has the same key, because the relationship between its data elements is many-to-many. For a given contract the quantity of an item may be divided into parts (QTYTBDEL), and each part is delivered on a different date (DEL-DATE). At the same time there may be many items that are delivered on one date. This is a many-to-many mapping, represented by a double-headed arrows in both direction.

IV. Data Base Design

Designing the data base will start with the requirements represented by the set of forms and reports described in section 3.2 and shown in Appendix B, and the assumptions stated in section 3.3. The design process given in section 2.2.2 will be applied to perform the following:

1. Determining the data relationships and the users' views.
2. Developing third normal form relations.
3. Drawing the conceptual model.
4. Mapping the conceptual model to the logical model.

4.1 Data Relationships and Users' Views

For each report the relationships between its data elements were determined taking into consideration the appropriate assumptions. A set of third normal form relations, representing the users' view for the given report, was developed, and the key of each relation was underlined. Determining the data relationships and the users' views for the first two reports in Appendix B are shown below in detail. The same steps are performed for all the other reports and the final results represented by the users' views are given in Appendix D.

1. Situation of Vendor's Contracts

   The data elements representing the entities of this report are:

   VEND-NO, VEND-NAME, ADDRESS,
   PHONE-NO, DATE, CONT-NO, ITEM-NO,
   ITEM-NAME, CONT-QTY, DEL-QTY, QTYUDEL,
   QTYTBDEL, DELDAT, TOTAL , PAID , REST

The relationships between these data elements are:

24. The quantity on hand (QTY-ON-HAND) is the quantity in the section at the date of the end of the report period.

25. Warehouses are audited once every six months. Also a section within a warehouse or the whole warehouse may be audited at any time, according to the management orders.

26. The audit result (AUDIT-RES) is the actual quantity of the item found in the section as a result of auditing.

27. PLUS/MINUS is the difference between the quantity on hand registered in the section records and the actual quantity found.

28. The total value of difference (T-V-DIFF) is recorded whether the difference is positive or negative.

29. In the "Addition/Withdrawal Transaction" the transaction code (X-CODE) can have only one of two values, 'A' for addition or 'W' for withdrawal. It is the only transaction that is recorded.

30. Inquiry transactions are not recorded.

leather clothes, and others.

14. A stock holder may be responsible for a number of sections within one warehouse.

15. The section number (SEC-NO) is unique and identifies the section within the warehouse and among all the sections in all warehouses. It consists of two characters, the first letter of the warehouse name and the serial number of the section in the warehouse.

16. The item number is unique. It consists of three digits representing the serial number of the item in the section concatenated with the section number.

17. Item name is not unique, many items may have the same name.

18. Raw materials are a special type of items.

19. Raw material number (RN-NO) uniquely identifies the raw material. It has the same structure as the item number.

20. Size number is unique within the set of sizes of the item.

21. The total raw material (TOTAL-RM), in the production order, is equal to the required quantity of the item (REQ-QTY) multiplied by the average raw material quantity used in producing one item unit (AVR-QTY/UNIT).

22. The "Total Weekly Transaction" report is a summary about all the transactions (additions and withdrawals) that occured in a section during one week.

23. The start quantity (START-QTY) is the quantity on hand in the section at the date of start of the report period (DSTRPT).

3.2.2 Batch and On-line Applications

Reports 4, 5, 6, 7, 8, 9, 10 and 11 can be considered as batch applications, and the others are on-line transactions. This is to be taken into consideration in the design and implementation of the data base.

3.3 Assumptions About Data

The assumptions about the requirements, represented by the set of reports given in Appendix B, are as follows:

1. The contract number is unique.
2. A contract may contain one or more items, but the contract is made with only one vendor.
3. The quantity of an item in the contract may be divided into parts, each part to be delivered by a specific date.
4. The vendor number is unique, but the vendor name is not unique.
5. If a vendor has more than one branch, only the address and telephone number of his central branch are kept in his records.
6. The factory name is unique.
7. Each factory has a separate building.
8. At a given point in time a factory has only one factory manager.
9. Machine number is unique within the factory in which the machine exist only. In order to identify the machine both the factory name and the machine number are used together.
10. The warehouse name is unique.
11. Each warehouse has a separate building.
12. At a given point in time a warehouse has only one manager.
13. Each warehouse consists of a number of sections, e.g., the clothes warehouse consists of the sections woolen clothes, cotton clothes,

report contains the quantity on hand and the date of last withdrawal. The report is sent to the depot's management and the logistics headquarters to cancel the item and sell the existing quantities or take any other decision.

11. Critical Items List

This report is prepared at any time an item's quantity reaches the critical quantity, which is the least amount of the item to be in the warehouse. The report is sent to the management to stop supply orders to subsidiary depots, and to take the necessary action for replenishing the warehouse.

12. Items Information

This report is printed at any time, and for any department at its request. It contains all the information about an item such as unit price, duration time, description, and specifications. It is used for different reasons in each department.

13. Addition Withdrawal Transaction

This form is printed whenever a quantity of an item is delivered to the warehouse, and also whenever a quantity is withdrawn from the warehouse. The transaction code (X-CODE) is 'A' for addition and 'W' for withdrawal.

14. Stock of Items

This report is printed for the management at its request. It shows the quantities on hand of the items asked for by the management.

number and name the report contains the quantity on hand at the beginning of the week, the sum of the added quantities, the sum of the withdrawn quantities, and the quantity on hand at the end of the week.

8. Audit Report

The sections within a warehouse are audited periodically, as well as for specific reasons. The audit report may be for all the items in the section, or for specific items. It contains the quantity on hand which is registered in the section reports, the actual quantity found in the section (AUDIT-RES), the difference (PLUS/MINUS), and the total value of the difference. This report is sent to both the depot's management and to the logistics headquarters.

9. Total Stock

This report is printed for all warehouses at the end of the fiscal year. It may be also printed at any time for specific warehouses according to the request of management. It contains for each item in the warehouse the quantity on hand and the warehouse capacity, which represents the maximum quantity of the item that can be stored in the warehouse.

10. Idle Items List

This report is prepared on a monthly basis. It contains the items for which no withdrawals have occured during the year ending in the date of the report. For each item the

V. Partial Implementation of the Data Base

The logical data base represented by the six relations of Appendix G. was implemented on INGRES, the relational DBMS available at AFIT. These six relations were named the production management data base, because they constitute a complete subset of relations that enable manipulating the data used in this field. The implemented data base can be used to generate the reports listed in Appendix B-3 and B-4, and any other unanticipated requirements concerned with the same data.

5.1 INGRES Environment (7,), (8)

As a relational DBMS, INGRES is relatively a simple system. Creating and maintaining a data base on INGRES is summarized in the following steps:

1. Creating the Data Base

This is done by using the following command to the UNIX shell

```
% Createdb  <database name>
```

Upon executing this command the user becomes the DBA, with all his powers.

Once the data base has been created the DBA should run the sysmod program using the command

```
% sysmod  <data base name>
```

This program converts the system relations to their best structure for use on INGRES.

2. Creating the Relations

There are two ways to create new relations in INGRES by using the commands "create" or "retrieve into". The "retrieve into" is used to form a new relation from one or more existing relations. The "create" is used to create a new relation with no tuples in it. This latter command has the format:

```
Create <relation name>  (attribute a =< field type & length,
                         attribute b =< field type & length,
                              .
                              .
                         attribute n =< field type & length)
```

INGRES limits a relation to no more than 49 domains with the tuple width limited to 498 bytes.

3. Loading the Data

Once a relation is created, there are two ways for inserting new data in it, by using the "append" or the "copy" command. The "append" is used to insert tuples one at a time. The "copy" is used for copying data from a UNIX file into a relation and vice versa. For example to copy data from a file into a relation, the command has the format:

```
copy <relation name >  (attribute a = <field type & length,
                        attribute b = <field type & length,
                             .
                             .
                        attribute n = <field type & length)
```

4. Choosing the Data Structure

Once the relations are created and the data is inserted, INGRES can process any query on the relations. In order to improve the speed at which INGRES can process queries, the DBA should specify the storage structure.

INGRES can store relations in three internal different structures, heap, isam, and hash. When a relation is first created it is created as a heap. The DBA may convert relations to any storage structure, according to his understanding to the nature of the requirements. This is done using the "modify" command. For example to convert a relation to isam the command has the following format:

```
modify < relation name >  to isam on key1, key2,...
```

5.2 Implementing the Data Base

The production management data base was created, the sysmod program was run, and the relations were created, as described in the previous section. The data was copied from UNIX files using the "copy" command.

The items and f-machines relations were converted to isam, because it will be usually required to retrieve specific tuples from these relation, and it will be also required to retrieve all the tuples at some other times. The other 4 relations were converted to hash, because it will be always required to retrieve certain tuples, and it will never be needed to retrieve all of them.

The structure of these 6 relations is given in Appendix H, showing the attributes of each relation, the field type and length of each attribute, and the defined keys.

5.3 Testing the Data Base

The implemented data base was tested by running some queries against it, and they gave the expected results. These queries were designed to retrieve the data required to prepare a production plan of an item to the clothing factory, as will be shown in Chapter VI. the queries and the retrieved data are listed in Appendix I.

VI. The Cutting Problem Application

Raw materials are delivered to the main depot's factories in standard units, such as cloth rolls, metal bars, and so. The standard units of a raw material have the same dimensions, i.e., they are of the same length and width. In order to produce usable items, amounts of the standard units are cut into smaller pieces with lengths suitable for fabricating the required items.

One of the major problems of the factories is the waste that results during the cutting process. Different cutting methods are applied in different factories, and the waste percentage differs according to the used method.

6.1 Cutting Methods

6.1.1 One Size Patterns

In this method the unit of raw material is cut into a number of pieces, all of the same length. This is applied in the clothing factories. In these factories an item is produced in different sizes, each of them requires a certain length of cloth, and is produced in a specific percentage of the whole production of the item.

The required quantities of the different sizes of the item are produced one at a time, by cutting a number of rolls into pieces with the length required for the produced size. The same process is repeated for all other sizes until the production is completed. No trial is made to find combinations of sizes to be cut together from one roll, because the rolls are usually very long, and the number of different combinations is very large.

Cutting the roll by this way, into pieces with the same length, yields a certain number of pieces, and usually a smaller piece is left at the end. This piece is considered as trim loss, or waste, because its length is less than that required for one unit of the required size. Using this cutting method causes a waste percentage of 5.7% of the total raw material used in the clothing factories.

6.1.2 Multisize Patterns

In this method the raw material unit is cut into pieces of different lengths. This is applied in the metal and woodworks factories, because the standard bars have very much smaller lengths as comparied to the cloth rolls. Accordingly, the workers, by their experience, can find combinations of the required lengths to be cut from the same bar.

Using multisize combinations decreases the waste at the beginning of production process. But as the production proceeds the required quantities of some lengths are completed, hence the lengths that still have to be cut decrease . Consequently the number of possible combination decreases, and the waste starts to increase.

At the end, usually a quantity of only one length is to be cut in order to complete the production. This quantity is cut separately, i.e., a number of bars is cut according to the one size pattern method. The waste percentage caused due to using this method is about 9.6%.

6.2 Solution Approach

The solution to the cutting problem is summarized in preparing a cutting plan before starting the cutting process. The plan should describe the set of cutting patterns to be used, and the number of standard raw material units to be cut according to each pattern.

These requirements can be achieved by using linear programming techniques. These techniques yield solutions that are considered optimal cutting plans. Solving a cutting problem by these techniques offers the optimal set of efficient cutting patterns, and the minimum number of raw material units to be cut according to each pattern. The solution guarantees producting the required quantities of all the desired lengths, and minimizing the used raw material units, which means minimizing the waste.

The two linear programming techniques used for solving such kind of problems are the simplex and the column generation techniques. Using any of these techniques requires formulating the problem as a linear program, and using the technique algorithm to solve this linear program.

6.2.1 The Simplex Method (11), (15)

This method is used when the problem size is small, i.e., the number of the different required lengths is small, and at the same time the raw material unit can be cut into a few number of pieces. In this case all the efficient cutting patterns could be listed without too much difficulty. The efficient patterns are used in formulating the problem as a linear program and solved using the simplex algorithm. The solution steps are described as follows:

1. Listing the Efficient Patterns

An efficient pattern is a cutting setup in which a number of pieces of one or more different length is cut from one raw material unit. All patterns of this type are determined and listed in a table.

2. Problem Formulation

The contents of the patterns' table are used in formulating the linear program according to the following general steps.

(1) Identify the unknown variables and represent them in terms of algebraic symbols. The unknown variables here ar the numbers of raw material units to be cut according to the efficient patterns.

(2) Identify all the constraints and express them as linear equations or inequalities. For each length there is an associated constraint, which limits the number of pieces of that length which will be cut in all patterns, to the required quantity.

(3) Identify the objective and represent it as a linear function of the unknown variables.

3. The Simplex Algorithm

The simplex method is an iterative procedure for solving linear programming problems. In summary the basic steps of the simplex method for solving a minimization problem are as follows:

(1) Start with an initial basic feasible solution and setup the initial tableau.

(2) Compute the relative coefficients of all the nonbasic variables. If all of them are nonnegative, the current solution is optimal. Otherwise select the nonbasic variable with the lowest coefficient to enter the basis.

(3) Apply the minimum ratio rule to determine the basic variable to leave the basis.

(4) Perform the pivot operation to get the new tableau and the new basic feasible solution.

(5) Return to step 2.

4. Modifying the Solution Steps

A major requirement of the simplex method is the availability of an initial basic feasible solution (IBFS). To find the IBFS one variable from each constraint is selected by inspection to be one of the basis in this intial solution.

In many practical problems this requirement may not be easily reached. The systematic way of getting a system of constraints with a basic feasible solution, when none is available by inspection, is the addition of artificial variables to the constraints. The problem is then solved by applying the two phase simplex algorithm, in which the simplex algorithm is applied twice, one time to solve the artificial problem, and the other to solve the original problem.

Solving a problem using the two phase algorithm is lengthy and time consuming. At the same time it can be avoided by a simple modification in listing the cutting patterns. In addition to listing all the efficient patterns, the set of one size patterns are also listed, even if they are not efficient. By this means the variables corresponding to these latter patterns constitute an IBFS to the problem. This modification enables the use of the simplex algorithm directly to solve the problem, and avoiding the use of the two phase algorithm.

Using the described technique is demonstrated through the following examples:

It is required to cut pieces of lengths 2, 3, and 4 feet in minimum amounts of 250, 60 and 180 pieces respectively, from standard rolls of 10 feet length.

1. Listing the Efficient and One Size Patterns

It is possible in this problem to determine all the efficient patterns. These patterns in addition to the one size patterns are listed in the following table:

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9
2 feet	5	0	0	3	3	2	1	1	6
3 feet	0	3	0	1	0	2	0	1	2
4 feet	0	0	2	0	1	0	2	1	1
Total length cut	10	9	8	9	10	10	10	9	10
Trim loss	0	1	2	1	0	0	0	1	0

Table 6.1. Cutting Patterns

Each column in the table represents a different pattern of cutting. For example, column 4 corresponds to cutting a standard roll into 3 pieces of length 2 feet and one piece of length 3 feet, which leaves a trim loss of one foot.

All columns describe patterns which are efficient except column 3, which is added to complete the set of one size patterns Y_1, Y_2, and Y_3. Column 3 is added to facilitate finding the IBFS and to avoid using the two phase simplex algorithm as described previously.

The ith cutting pattern is represented by a symbol Y_i, and the number of rolls that will be cut according to this pattern will be represented by the symbol x_i.

2. Problem Formulation

The objective of solving this problem is to meet the demands and minimize the number of standard rolls that are cut, so the program to be studied is:

Minimize $z = x_1 + x_2 + \ldots\ldots + x_9$

subjected to

$$\begin{aligned}
5x_1 \quad &+ 3x_4 + 3x_5 + 2x_6 + x_7 + x_8 &\geq 250\\
3x_2 \quad &+ x_4 + 2x_6 + x_8 + 2x_9 &\geq 60\\
2x_3 \quad &+ x_5 + 2x_7 + x_8 + x_9 &\geq 180
\end{aligned}$$

and $x_i \geq 0$ for $i = 1, \ldots, 8$

Each nontrivial constraint is derived by adding the number of pieces of a particular length, that would result from x_i of each type of cut.

This program is transformed to the standard form by subtracting the nonnegative slack variable s_1, s_2, s_3 as follows:

Minimize $z = x_1 + x_2 + \ldots + x_9$

subjected to

$$\begin{aligned}
5x_1 \quad &+ 3x_4 + 3x_5 + 2x_6 + x_7 + x_8 - s_1 &= 250\\
3x_2 \quad &+ x_4 + 2x_6 + x_8 + 2x_9 - s_2 &= 60\\
2x_3 \quad &+ x_5 + 2x_7 + x_8 + x_9 - s_3 &= 180
\end{aligned}$$

and $x_i \geq 0$ for $i = 1, \ldots, 8$

$s_j \geq 0$ for $j = 1, 2, 3$

The nonnegative slack s_1 represents the amount of pieces with 2 feet length, by which pieces of this length cut in all setups, will exceed 250 in the final solution.

For example if these pieces in the final solution equal 270, then s_1 must equal 20 in order for the first equation to hold. S_2 and s_3 have the same meaning w.r.t. the other two lengths.

3. Solving the Problem

According to the simplex algorithm the problem is represented in tableau format by the following tableau 0.

Tableau 0

Basis	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	s_1	s_2	s_3	
x_1	$\textcircled{5}$	0	0	3	3	2	1	1	0	-1	0	0	250
x_2	0	$\textcircled{3}$	0	1	0	2	0	1	2	0	-1	0	60
x_3	0	0	$\textcircled{2}$	0	1	0	2	1	1	0	0	-1	180
	1	1	1	1	1	1	1	1	1	0	0	0	z

The first 3 columns are divided by the circled entries to convert them to unit vectors, then the obtained rows are subtracted from the objective row to make the coefficients of the basic variables in this row zeros, and tableau 1 is formed.

Tableau 1

Basis	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	s_1	s_2	s_3	
x_1	1	0	0	3/5	3/5	2/5	1/5	1/5	0	-1/5	0	0	50
x_2	0	1	0	1/3	0	2/3	0	1/3	2/3	0	-1/3	0	20
x_3	0	0	1	0	1/2	0	1	1/2	1/2	0	0	-1/2	90
	0	0	0	1/15	-1/10	-1/15	-1/5	-1/30	-5/30	1/5	1/3	1/2	Z = 160

This tableau represents the IBFS with the objective value of 160, and the basic variables x_1, x_2, and x_3, which corresponds to the first three patterns y_1, y_2, and y_3, having the values 50, 20, and 90 respectively.

Pivoting on the circled entry gives tableau 2

Tableau 2

Basis	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	s_1	s_2	s_3	
x_1	1	0	-1/5	3/5	5/10	2/5	0	1/10	-1/10	-1/5	0	1/10	32
x_2	0	1	0	1/3	0	2/3	0	1/3	2/3	0	-1/3	0	20
x_7	0	0	1	0	1/2	0	1	1/2	1/2	0	0	-1/2	90
	0	0	1/3	1/15	0	-1/15	0	1/15	-1/15	1/15	1/3	2/5	Z = 142

repeating the same steps and pivoting on the circled entry tableau 3 is obtained

Tableau 3

Basis	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	s_1	s_2	s_3	
x_1	1	-3/5	-1/5	2/5	5/10	0	0	-1/10	-5/10	-1/5	1/5	1/10	20
x_6	0	3/2	0	1/2	0	1	0	1/2	0	0	-1/2	0	30
x_7	0	0	1	0	1/2	0	1	1/2	1/2	0	0	-1/2	90
	0	1/10	1/5	1/10	0	0	0	1/10	0	1/5	9/30	2/5	$z = 140$

This tableau is optimal, because all entries in the objective row are nonnegative, which means that no further improvement can be made in the objective value. In the optimal solution, 20, 30, and 90 rolls are to be cut according to patterns y_1, y_2, and y_3 respectively. All these patterns yield zero trim loss as described in the table of the efficient patterns.

6.2.2 The Column Generation Method (15)

The column generation, or the implicit tableau, method is used whenever a cutting problem is so complex that it is impossible to enumerate all the efficient patterns. In this method the complete enumeration is replaced by partial enumeration.

The major reason behind developing this technique is that step 2 in the simplex algorithm describes only one way to selecting the pivot columns, that is selecting the one with the lowest negative coefficient. But this is done only to enhance the rate of improving the objective value. Actually any negative coefficient determines a suitable pivot column.

This means that it is possible to start the initial tableau with the one size patterns alone. The efficient patterns are then determined, or generated, one by one, and after generating each pattern it is examined, if it gives a negative coefficient then it is entered in the basis replacing an old pattern, otherwise it is discarded, and so on.

At a certain stage in the solution it appears that it is not necessary to determine all the efficient patterns that have not yet been determined, but only a subset of them that satisfy the condition specified by the typical column of the current tableau.

The column generation technique is demonstrated through the following example:

The standard roll is 130 meters long, and the required pieces are at least 30 piece of length 40 meters, at least 60 pieces of length 35 meters, and at least 60 pieces of length 25 meters.

1. The Typical Pattern

An implicit pattern $y_i = \begin{pmatrix} u \\ v \\ w \end{pmatrix}$ that represents the pattern in which u pieces of length 40, v pieces of length 35, and w pieces of length 25, is considered to denote any pattern, not necessarily efficient.

2. Problem Formulation

The problem is formulated as an implicit linear program as follows:

Minimize $x = x_1 + x_2 + \ldots + x_i \ldots + x_n$

subjected to

$$x_1 y_1 + x_2 y_2 + \ldots + x_i y_i + \ldots + x_n y_n \geq \begin{pmatrix} 30 \\ 60 \\ 60 \end{pmatrix}$$

and $x_i \geq 0$ for $i = 1, \ldots, n$

6. SECTION	The warehouse consists of a number of sections. Each section has a symbol of two characters, the first letter of the warehouse name, and the serial number of the section within the warehouse. For example the woolen clothes section has the symbol C1, and the cotton clothes section has the symbol C2.
7. VENDOR	A vendor is identified by his number and name. The depot's management maintains a list of the vendors, to be referenced in case of any information about a vendor is needed.
8. WAREHOUSE	Warehouses within the depot are classified into two groups, finished items warehouses and raw materials warehouses. Each warehouse has a name derived from the name of the items stored in it.

Appendix A

List of Entities

Entity	Description
1. CONTRACT	Contracts have serial numbers starting from the beginning of the fiscal year. The contract number consists of two parts, the serial number, and the year e.g. (01234/84). The contract contains information about the vendor, vendor no, name, address, and phone no. Also it contains information about the item(s), item no., name, quantity, dates of delivery, and total price. A contract may contain any number of items, but the contract is made with only one vendor.
2. ITEM	The item is any finished product delivered to the warehouses directly by vendors, or as a fabricated product in the main depot factories. It is the most important entity within the depot, all the functions and processes are concerned with it in some means. It also has relationships with all other entities as seen in the entity relationship chart (Fig 3.1).
3. FACTORY	The factory is a production unit within the main depot. Each factory is responsible of producing certain types of items. The production process starts by receiving a production order from the depot's management. The factory withdraws the raw material from specific warehouse, and delivers its production to the finished items warehouses.
4. MACHINE	A machine is distinguished with its number and name. A factory may have many machines of the same type, but they have different numbers. A machine may produce many items, but a given item is produced by one machine types.
5. RAW MATERIAL	Raw materials are a special type of items, delivered to the depot by vendors and stored in special warehouses. They are actually intermediate items, not used as they are and not supplied to the subsidiary depots. It was found that it would be better to define raw material as a separate entity, because it is involved in special processes.

16. Ulman, J.D. Principles of Data Base Systems. Rockville MD: Computer Science Press, Inc., 1982.

17. Zaniolo, C. "On the Design of Relational Data Base Schemata," ACM Trans. Database Syst., 6,1: 1-47 (March 1981).

Bibliography

1. Atre, S. Data Base Structured Techniques for Design, Performance and Management. New York: John Wiley & Sons, Inc., 1980.

2. Bernstein, P.A. "Synthesizing Third Normal Form Relations from Functional Dependencies," ACM Trans. Database Syst., 1,4: 277-298 (Dec 1976).

3. Cardenas, A.F. Data Base Management Systems. Boston: Allyn & Bacon, Inc., 1979.

4. Codd, E.F., "A Relational Model of Data for Large Shared Data Banks," ACM Trans. Comm., 13,6: 377-387 (June 1970).

5. Date, C.J. An Introduction to Data Base Systems. Reading, MA: Addison-Wesley Publication Co., 1982.

6. Date, N. and Orshalick, D. Introduction to Pascal and Structured Design. Lexington MA: D.C. Heath and Company, 1983.

7. Electronics Research Laboratory, College of Engineering. Memorandom of A Tutorial on INGRES. University of California, Berkely, 15 Dec 77.

8. Electronics Research Laboratory, College of Engineering. Memorandum of Creating and Maintaining a Data Base Using INGRES. University of California, Berkely, 16 Dec 77.

9. Fagin, R. "The Decomposition Versus Synthetic Approach to Relational Database Design," Proc. Third Int. Conf. Very Large Data Bases. 441-446 Tokyo, Japan, Oct 1977.

10. Kroenke, D. Data Base Processing. Chicago: Science Research Associates, Inc., 1983.

11. Levin, R. and Kirkpatrik, C. Quantitative Approach to Management. New York: McGraw-Hill, Inc., 1978.

12. Martin, J. Managing the Data Base Environment. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1983.

13. Martin, J. Principles of Data Base Management. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1976.

14. Simmons, D. Linear Programming for Operations Research. San Francisco: Holden Day, Inc., 1972.

15. Strum, J.E. Introduction to Linear Programming. San Francisco: Holden Day, Inc., 1972.

7. Application programs, for producing reports and any other information requirements, should be developed.
8. Finally, it is necessary to stress on implementing the data base step-by-step, to enable the users of accepting the new techniques gradually, and to ensure success.

5. The software system developed to solve cutting problems by the column generation technique has many advantages over other systems based on the simplex algorithm. In addition to giving the same optimal solutions, it has the minimum space and time requirements.

7.2 Recommendations

Based on the facts and assumptions about the characteristics of the environment, and the conclusions from this study, the following recommendations are proposed.

1. The relational DBMSs available commercially should be studied, to select one which is efficient and compatible with the existing hardware and operating system.
2. Interviews should be conducted with management and users in the areas of inventory control and production management, to justify the names of the data elements before implementing the data base.
3. Top-down analysis should be completed to obtain a more detailed entity relationship chart that represents the whole depot. This chart will be divided into a number of subject data bases.
4. Recommendation 1, 2, and 3 may start in parallel, by three different teams under the supervision of the DBA.
5. The data base implementation should start with the developed inventory control and production management data base, because they are the most important areas in the depot.
6. The software system developed to prepare cutting plans should be applied, even before implementing the data base.

VII. Conclusions and Recommendations

7.1 Conclusions

In this study a relational data base for inventory control and production management was developed. Several important conclusions were reached during the course of study. These conclusions are:

1. The relational model was found to be the most suitable one among the three main data base models for application in the main depot. This model is highly flexible and easy to use, and by these characteristics it is suitable for such environment with no background in data base.
2. It was found that it is better in this environment to start with a small data base project, and when it works the data base can be expanded one step at a time. The first project was chosen to support the two important areas of inventory control and production management.
3. The conceptual model was developed in a canonical form. This is due to fact that themain depot has not yet purchased a DBMS, and that there may be restrictions on the suitable DBMS type, such as the existing hardware and operating system. In this case the conceptual model can be mapped to the appropriate model.
4. It appeared during the implementation stage that INGRES, as a relational DBMS, is a simple system. It provides a great deal of flexibility for the user in data base creation and loading. In addition the English like query language (QUEL) facilitates using the system by nonprogramming users.

(3) Compute the total raw material cost and some other required information.

The software system is tested and documented as listed in Appendix J. A sample problem and its solution are given in Appendix K. This problem was solved prior to working in this thesis using a commerical simplex package on an IRIS 50 computer in Egypt, and the run took 2 hours and 14 minutes. The same solution was obtained several times using the developed system on the VAX 11 in AFIT and the runs took an average time 6 minutes. Although this comparison must be made under the same conditions and on the same machine, but this approximate results show great difference between the developed system and the simplex package.

It is still to say that this system, by its current structure, could be used by feeding it with the data retrieved from the data base separately. But upon choosing the DBMS that will be used in the main depot, the system should be translated to the host language, to allow the addition of some embedded queries for data retrieval. In this case the input will be limited to the item number and the required quantity.

6.3 Computerizing the Column Generation Technique

The advantages of the column generation technique over the simplex constituted a major motivation towards automating this technique, because up to the author's knowledge there are no commercial packages for it.

An interactive user friendly software system was developed, according to the standards of software engineering and structured programming, and coded in PASCAL. The system solves cutting problems according to the prescribed technique, and contain some additional processes that minimize the space and time requirements. These processes are:

1. Upon generating a pattern that is suitable to enter the basis, it replaces the leaving pattern in its column, since this latter pattern is never required again. This allowed using static arrays of maximum size N x N, where N is the number of required lengths, for storing the patterns.
2. The input data is specified to be entered to the system with the lengths arranged in descending order. This arrangement made it possible to include some conditions that decrease the generated patterns, without losing any pattern that could improve the solution.
3. A procedure for final calculations is included to perform the following steps.
   (1) Change the solution to be in the form of integer numbers, instead of the obtained real numbers, without affecting the solution optimality.
   (2) Compute the trim loss in each optimal pattern, and the total trim loss percentage in both the initial and the final solution, for comparison purposes.

hence h_3 will surely be positive if $u + v + w < 4$ then it appears that it is needed only to examine the subset of patterns, for which $u + v + w > 4$. These patterns are listed in the following table:

	solution (u, v, w)	value of h_3
w = 0	no relevant solution	
w = 1	no relevant solution	
w = 2	1, 1, 2	0
	0, 2, 2	positive
w = 3	1, 0, 3	positive
	0, 1, 3	positive
w = 4	0, 0, 4	positive
w = 5	0, 0, 5	0

Table 6.2. Efficient Patterns

If the h_3 value for any of these patterns were negative, then the pattern would have been entered in tableau 3 and the solution would have been continued.

It still remains to see whether the columns of tableau 3 corresponding to the slack variables need to be recovered or not. To recover the body of the s_1 column the values $u = -1$, $v = 0$, and $w = 0$, are substituted into the body of the x_1 column. To recover the coefficient of s_1 this substitution is made into the general expression for $h_3 - 1$ (not into h_3), which gives a value of 3/10. By similar reasoning it can be seen that the coefficients of s_2 and s_3 are also positive. If any coefficient were negative, then it is necessary to recover its column completely and continue pivoting.

One solution which makes h_2 negative is u = 2, v = 0, and w = 2, for which $h_2 = 1/15$. Entering the new column by substituting the values of u, v, and w in the typical column, and pivoting gives the following tableau.

Tableau 3

Basis	x_1	x_2	x_3	...	x_i	x_4	x_5	
x_5	2/3	0	0		1/2u	0	1	15
x_4	0	1	0		1/3v	1	0	20
x_3	-3/5	-1/5	1		1/15w - 1/15v - 1/5u	0	0	2
	14/15	1/5	0		h_3	0	0	Z = 37

The value of h_3 is given by

$$h_3 = -3/10u - 4/15v - 1/5w + 1$$

multiplying by 30

$$30h_3 = -(gu + 8v + 6w) + 30$$

and since each pattern must satisfy the inequality

$$8u + 7v + 5w \leq 26$$

then

$$30h_3 = -(8u + 7v + 5w) - (u + v + w) + 30$$
$$= -(u + v + w) + 4$$

The column representing this pattern is then entered in tableau 1 under the heading x_4 and this new column is formed by substituing $u = 0$, $v = 3$, and $w = 1$ into the general x_i column as indicated in the following tableau:

Tableau 1'

Basis	x_1	x_2	x_3	...	x_i	x_4	s_1	s_2	s_3	
x_1	1	0	0		1/3u	0	-1/3	0	0	10
x_2	0	1	0		1/3v	$\textcircled{1}$	0	-1/3	0	20
x_3	0	0	1		1/5w	1/5	0	0	-1/5	12
	0	0	0		h_1	-1/5	1/3	1/3	1/5	z = 42

After pivoting on the circled entry in the x_4 column tableau 2 is obtained

Tableau 2

Basis	x_1	x_2	x_3	..	x_i	x_4	...	
x_1	1	0	0		1/3u	0		10
x_2	0	1	0		1/3v	1		20
x_3	0	-1/5	1		1/5w - 1/15v	0		8
	0	1/5	0		h_2	0		z = 38

The new x_i column represents the general column of tableau 2, and it is clear that

$$h_2 = -1/3u - 4/15v - 1/5w + 1$$

Tableau 1

Basis	x_1	x_2	x_3	$\cdots$	x_i	$\cdots$	s_1	s_2	s_3	
x_1	1	0	0		1/3u		-1/3	0	0	10
x_2	0	1	0		1/3v		0	-1/3	0	20
x_3	0	0	1		1/5w		0	0	-1/5	12
	0	0	0		h_1		1/3	1/3	1/5	Z = 42

The coefficient of the typical x_i in the objective row in tableau 1 is a function of u_1, v_1 and w, whose value is easily seen to be

$$h_1 = -1/3u - 1/3v - 1/5w + 1$$

Clearly the choice of a pivot column in tableau 1 can be reduced to finding values for u, v, and w that make h_1 negative. For the moment any pattern that satisfies this condition may be considered, to proceed in demonstrating the technique. But later on the problem of finding such patterns in a successive and organized manner will be dealt with, when programming the technique to be handled by the computer.

It is easily seen that $h_1 = 0$ for any cutting pattern corresponding to a variable which is currently basic. But for the pattern (u = 0, v = 3, w = 1), $h_1 = -1/5$. This means that using this pattern will improve the objective value, thus some old pattern should be replaced by this new pattern.

$$y_4 = \begin{pmatrix} 0 \\ 3 \\ 1 \end{pmatrix}$$

The components of the typical pattern must satisfy the inequality

$$40u + 35v + 25w \leq 130$$

or equivalently

$$8u + 7v + 5w \leq 26$$

From this inequality the following pure, but not necessarily efficient patterns can be deduced.

$$y_1 = \begin{pmatrix} 3 \\ 0 \\ 0 \end{pmatrix}, \quad y_2 = \begin{pmatrix} 0 \\ 3 \\ 0 \end{pmatrix}, \quad y_3 = \begin{pmatrix} 0 \\ 0 \\ 5 \end{pmatrix}$$

These 3 pure patterns in addition to the typical pattern, and three columns representing the slack variable are listed in tableau 0, and the simplex algorithm is applied to solve the problem.

Tableau 0

Basis	x_1	x_2	x_3	. . .	x_i	. . .	s_1	s_2	s_3	
x_1	$\bigcirc\!\!\!\!3$	0	0		u		-1	0	0	30
x_2	0	$\bigcirc\!\!\!\!3$	0		v		0	-1	0	60
x_3	0	0	$\bigcirc\!\!\!\!5$		w		0	0	-1	60
	1	1	1		1		0	0	0	Z

This tableau is not yet adjusted, but the first three columns could be converted into unit vectors by pivoting in any order, on the circled entries in row i, column i, for i = 1, 2, 3. Thus the initial basic feasible solution is obtained as shown in the following tableau.

Appen [illegible]

Requirements

Situation of Vendor's Contracts

VEND-NO	:	1242
VEND-NAME	:	John Smith
ADDRESS	:	5841 Grass Av. Cairo
PHONE NO	:	2561[illegible]3
DATE	:	July 31, 1984

CONT-NO	ITEM-NO	CONT-QTY	DEL-QTY	QTYUDEL	QTYTBDEL	DEL-DATE	TOTAL	PAID	REST
1234/84	110C1	100,000	50,000	50,000	25,000	08/10/84	1,500,000	750,000	750,000
					25,000	09/12/84			
1462/84	141C2	20,000	5,000	15,000	15,000	10/09/84	200,000	50,000	150,000

Situation of Item's Contracts

ITEM-NO	:	110C1
ITEM-NAME	:	Battle dress
DATE	:	July 20, 1984

CONT-NO	VEND-NO	VEND-NAME	CONT-QTY	DEL-QTY	QTYUDEL	QTYTBDEL	DEL-DATE
1123/84	1242	John Smith	100,000	50,000	50,000	25,000	08/10/84
						25,000	09/12/84
2345/84	1135	James Cook	30,000		30,000	30,000	11/11/84

T-QTYUDEL : 80,000

Vendors List of Item

ITEM-NO : 110C1

ITEM-NAME : Battle dress

DATE : July 31, 1984

VEND-NO	VED-NAME	ADDRESS	PHONE-NO	VEND-CAP
1135	James Cook	2341 Hickam Dr	230144	200,000
1242	John Smith	5861 Grass Av.	256123	500,000
2567	Thomas Rex	2600 Grass Av.	267179	100,000

Weekly Machine Situation

FACT-NAME : Clothes

FACT-MGR : James White

BUILD-NO : 224

DATE : June 31, 1984

MACH-NO	MACH-NAME	STATUS	DATE-IDLE
122	Sewing Machine	W	
123	Sewing Machine	I	06/28/84

Production Data of Item

ITEM-NO	: 150C1	RM-NO	270R2
ITEM-NAME	: Long sleeve shirt	RM-NAME	Cotton drell

SIZE-NO	RM-QTY	PRODUCTION %
1	2.90	2
2	2.81	12
3	2.77	18
4	2.64	25
5	2.61	31
6	2.56	12

Production Order

FACT-NAME	:	Clothes
BUILD-NO	:	224
ITEM-NO	:	150C2
ITEM-NAME	:	Long sleeve shirt
REQ-QTY	:	20,000
RM-NO	:	270R2
RM-NAME	:	Cotton drell
RM-UNIT	:	Meter
RUNIT-PRICE	:	3.10
AVR-RM-UNIT	:	2.6701
TOTAL-RM	:	53,402 meter

Total Weekly Transactions

WH-NAME	:	Clothes
WH-MGR	:	James Brown
BUILD-NO	:	125
SEC-NO	:	C1
SH-NAME	:	John White
DSTRPT	:	July 16, 1984
DENDRPT	:	July 20, 1984

ITEM-NO	ITEM-NAME	START-QTY	ADDED-QTY	NO-ADD	WITHD-QTY	NO-WITHD	QTY-ON-HAND
110C1	Battle dress	125,000	20,000	4	25,000	22	110,000
115C1	Blue cap	76,200			15,100	10	61,100

Audit Report

WH-NAME	: Clothes
WH-MGR	: James Brown
BUILD-NO	: 125
SEC-NO	: C1
SH-NAME	: John White
DATE	: July 31, 1984

ITEM-NO	ITEM-NAME	QTY-ON-HAND	AUDIT-RES	PLUS/MINUS	UNIT-PRICE	T-V-DIFF
110C1	Battle dress	120,500	120,490	-20	40.20	$ 420
115C1	Blue cap	66,000	66,000		8.10	

Total Stock

WH-NAME	:	Clothes
WH-MGR	:	James Brown
BUILD-NO	:	125
SEC-NO	:	C1
SH-NAME	:	John White
DATE	:	July 25, 1984

ITEM-NO	ITEM-NAME	WH-CAP	QTY-ON-HAND
110C1	Battle dress	500,000	220,000
115C1	Blue cap	1,000,000	570,000

Idle Items List

WH-NAME	:	Clothes
WH-MGR	:	James Brown
BUILD-NO	:	125
SEC-NO	:	C2
SH-NAME	:	Ted Green
DATE	:	July 20, 1984

ITEM-NO	ITEM-NAME	QTY-ON-HAND	DLW
150C2	Long sleeve shirt	10,000	July 20, 1983

Critical Items List

WH-NAME	:	Clothes
WH-MGR	:	James Brown
BUILD-NO	:	125
SEC-NO	:	C1
SH-NAME	:	John White
DATE	:	July 10, 1984

ITEM-NO	ITEM-NAME	CRL-QTY	QTY-ON-HAND
110C1	Battle dress	20,000	20,100
115C1	Blue cap	10,000	10,200

Item Information

DATE	:	July 15, 1984
ITEM-NO	:	115C1
ITEM-NAME	:	Blanket
ITEM UNIT	:	One
UNIT-PRICE	:	20.50
DUR-TIME	:	3 years
DESC	:	Rectangular blanket, Dimensions 220x180 cm. Color Gray
SPECS	:	Material 50% wool, 50% cotton Edges 3cm gray satin

Addition/Withdrawal Transaction

SEC-NO	:	C1
SH-NAME	:	John White
DATE	:	July 31, 1984
X-CODE	:	A (Addition)

ITEM-NO	ITEM-NAME	ITEM-UNIT	QTYWD
110C1	Battle dress	by count	10,000
155C1	Gray stocking	by pair	25,500

Stock of Items

WH-NAME	:	Clothes
WH-MGR	:	James Brown
BUILD-NO	:	125
SEC-NO	:	C1
SH-NAME	:	John WHite
DATE	:	May 20, 1984

ITEM-NO	ITEM-NAME	QTY-ON-HAND
110C1	Battle dress	35,600
115C1	Blue cap	20,250

Appendix C

List of Data Elements

	Data Elements	Description	Size
1.	ADDED-QTY	The quantity of an item received by the warehouse for replinishment.	7
2.	ADDRESS	The address of the main branch of the vendor.	30
3.	AUDIT-RES	The actual quantity of an item found in a warehouse as a result of auditing.	5
4.	AVR-QTY/UNIT	The average quantity of the main raw material required to produce one unit of the item.	6
5.	BUILD-NO	The building number uniquely identifies the particular building.	3
6.	CONT-NO	Contract number uniquely identifies the contract. Numbers are serial and start at the beginning of the fiscal year. Example (1234/84).	7
7.	CONT-QTY	Quantity of the item registered in the contract.	7
8.	CRL-QTY	Critical quantity represents the least amount of an item to be in the warehouse.	5
9.	DATE	The date of preparing the report. It is a data element in all reports. Form (DD/MM/YY).	8
10.	DATE-IDLE	The date in which a machine started to halt.	8
11.	DEL-DATE	Delivery date registered in the contract for delivering the item to the depot.	8
12.	DEL-QTY	Quantity of the item delivered to the depot by a vendor.	5
13.	DENDRPT	Date of end of the total weekly transactions report.	8

	Data Elements	Description	Size
14.	DESC	Description of the physical properties of the item.	120
15.	DLW	The last date in which a quantity of an item was withdrawn from the warehouse.	8
16.	DSTRPT	Date of start of the total weekly transactions report.	8
17.	DUR-TIME	Duration time is the minimum time for an item to stay in a condition suitable for use, counted by years.	2
18.	FACT-MGR	Name of the factory manager.	30
19.	FACT-NAME	The factory name uniquely identifies the particular factory.	30
20.	ITEM-NAME	The standard name of the item. Different items may have the same name.	30
21.	ITEM-NO	The item number uniquely identifies the item. Items may have similar names, but different numbers. The item number consists of 3 digits denoting the serial number of the item in the specific section of the warehouse, the first letter of the warehouse name, and one digit denoting the specific section within the warehouse. Example (110C1).	5
22.	ITEM-UNIT	Each item has a specific unit, e.g., a pair of shows, or a dozen of stockings.	8
23.	MACH-NAME	The standard machine name. Different machines may have the same name.	40
24.	MACH-NO	The machine number uniquely identifies the machine within the factory.	4
25.	NO-ADD	The number of addition transactions occured in the section during one week.	2
26.	NO-WITHD	The number of withdrawal transactions occured in the section during one week.	3

	Data Elements	Description	Size
27.	PAID	The part of the contract value paid to the vendor until the date of the report.	6
28.	PLUS/MINUS	The difference between the quantity on hand of the item as it is registered in the warehouse records, and the actual quantity found as a result of auditing.	5
29.	PHONE-NO	The vendors telephone number.	6
30.	PRODUCTION %	The percentage of each size of the item in the whole production.	7
31.	QTY-ON-HAND	The quantity of the item as it is registered in the warehouse records.	5
32.	QTYTBDEL	The quantity of the item that should be delivered to the depot by the vendor at a specific date.	5
33.	QTYUDEL	The total quantity under delivery of an item. It is the sum of the quantities to be delivered (QTYTBDEL).	5
34.	QTYWD	Quantity withdrawn from or delivered to the warehouse section in one transaction.	5
35.	REQ-QTY	The total amount of the item to be produced, as specified in the production order.	5
36.	REST	The part of the contract value that is not paid to the vendor until the date of the report.	6
37.	RM-NAME	The standard name of the essential raw material used in fabricating the item. Different raw materials may have the same name.	30
38.	RM-NO	The raw material number uniquely identifies the raw material. It has the same structure as the item number.	5
39.	RM-QTY	The quantity of the essential raw material required to produce one unit of the item.	6

	Data Elements	Description	Size
40.	RM-UNIT	The measuring unit of the raw material, e.g. meter for cloth, and square foot for leather.	15
41.	RUNIT-PRICE	The price of one measuring unit of the raw material in Egyptian pounds.	5
42.	SEC-NO	The section number uniquely identifies the section within the warehouse, and within the whole depot. It consists of the first letter of the warehouse name and the serial number of the section within the warehouse.	2
43.	SH-NAME	The name of the stockholder of the section.	30
44.	SIZE-NO	The number of the particular size of the item.	
45.	SPECS	The standard specifications of the item.	200
46.	START-QTY	Quantity of the item in the warehouse at the beginning of the week, for weekly transactions report.	7
47.	STATUS	The status of a machine, one letter, W for working and I for idle and needs repairs.	1
48.	TOTAL	The toal value of the contract in Egyptian pounds.	6
49.	TOTAL-RM	The total amount of the essential raw material required to produce the quantity of the item specified in the production order.	9
50.	T-QTUDEL	The total quantity under delivery of the item, in all the contracts and for all the vendors.	7
51.	T-V-DIFF	The total value of the item quantity found as difference between the quantity on hand the actual audit result.	6

	Data Elements	Description	Size
52.	UNIT-PRICE	The price of one unit of the item in Egyptian pounds.	4
53.	VEND-CAP	Vendor capacity is the maximum amount of an item the vendor can contract on.	7
54.	VEND-NAME	The vendor's name. Different vendors may have the same name.	30
55.	VEND-NO	The vendor number uniquely identifies the vendor.	4
56.	WH-CAP	Warehouse capacity is the maximum amount of the item that can be stored in the warehouse.	7
57.	WH-MGR	Name of the warehouse manager	30
58.	WH-NAME	The warehouse name is derived from the name of the group of items stored in it e.g. clothes warehouse, or furniture warehouse.	5
59.	WITHD-QTY	The withdrawn quantity of the item from the section during one week, for weekly transactions report.	6
60.	X-CODE	The transaction code. One letter 'W' for withdrawal and 'A' for addition.	1

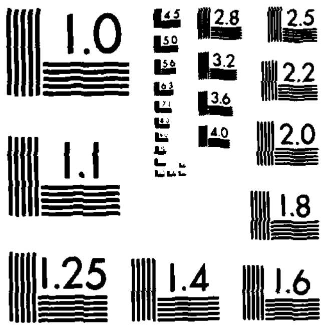
AD-A153 042 DEVELOPMENT OF A CONSOLIDATED DATA BASE FOR STOCK CONTROL AND PRODUCTION..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. A I EL DEIHI 2/2

UNCLASSIFIED DEC 84 AFIT/GCS/MATH/84D-3 F/G 9/2 NL

END

FILMED

DTIC


1.0
4.5
5.0
5.6
6.3
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

Appendix D

List of the Individual User's Views

1. Situation of Vendor's Contracts

(1) VEND-NO <---> VEND-NAME,ADDRESS,PHONE-NO
(2) CONT-NO <<---> TOTAL ,PAID ,REST
(3) ITEM-NO <<---> ITEM-NAME
(4) CONT-NO,ITEM-NO <<---> CONT-QTY,DEL-QTY,QTYUDEL
(5) CONT-NO,ITEM-NO,DEL-DATE <<---> QTYTBDEL

2. Situation of Item's Contracts

(6) ITEM-NO <<---> ITEM-NAME,T-QTYUDEL
(7) CONT-NO <<---> VEND-NO
(8) VEND-NO <<---> VEND-NAME
(9) CONT-NO,ITEM-NO <<---> CONT-QTY,DEL-QTY,QTYUDEL
(10) CONT-NO,ITEM-NO,DEL-DATE <<---> QTYUDEL

3. Vendor's List of Item

(11) ITEM-NO <<---> ITEM-NAME
(12) VEND-NO <<---> VEND-NAME,ADDRESS,PHONE-NO,
(13) ITEM-NO,VEND-NO <<---> VEND-CAP

4. Weekly Machine Situation

(14) FACT-NAME <---> FACT-MGR,BUILD-NO
(15) FACT-NAME,MACH-NO <<---> MACH-NAME,STATUS,DATE-IDLE

5. Production DATA of Item

(16) ITEM-NO <<---> ITEM-NAME
(17) RM-NO <<---> RM-NAME
(18) ITEM-NO,SIZE-NO <---> RM-QTY,PRODUCTION%

6. Production Order

(19) FACT-NAME <---> BUILD-NO
(20) ITEM-NO <<---> ITEM-NAME,FACT-NAME
(21) RM-NO <<---> RM-NAME,RM-UNIT,RUNIT-PRICE
(22) ITEM-NO,RM-NO <<---> AVR-QTY-UNIT

7. Total Weekly Transactions

(23) WH-NAME <---> WH-MGR,BUILD-NO
(24) SEC-NO <<---> SH-NAME
(25) ITEM-NO <<---> ITEM-NAME,QTY-ON-HAND
(26) ITEM-NO,DSTRPT,DENRPT <<---> START-QTY,ADDED-QTY,NO-ADD
WITHD-QTY,NO-WITHD

8. Audit Report

(27) WH-NAME <——> WH-MGR,BUILD-NO

(28) Sec-NO<<——> SH-NAME

(29) ITEM-NO <<——> ITEM-NAME,UNIT-PRICE,QTY-ON-HAND

(30) ITEM-NO,DATE <<——> AUDIT-RES,PLUS/MINUS,T-V-DIFF

9. Total Stock

(31) WH-NAME <——> WH-MGR,BUILD-NO

(32) SEC-NO <<——> SH-NAME

(33) ITEM-NO <<——> ITEM-NAME,WH-CAP,QTY-ON-HAND

10. Idle Items List

(34) WH-NAME <——> WH-MGR,BUILD-NO

(35) SEC-NO <<——> SH-NAME

(36) ITEM-NO <<——> ITEM-NAME,QTY-ON-HAND,DLW

11. Critical Items List

(37) WH-NAME <——> WH-MGR,BUILD-NO

(38) SEC-NO <<——> SH-NAME

(39) ITEM-NO <<——> ITEM-NAME,CRL-QTY,QTY-ON-HAND

12. Item Information

(40) ITEM-NO <<——> ITEM-NAME,ITEM-UNIT,UNIT-PRICE,DUR-TIME,
DESC,SPECS

13. Addition/Withdrawal Transaction

(41) SEC-NO <<——> SH-NAME

(42) ITEM-NO <<——> ITEM-NAME,ITEM-UNIT

(43) ITEM-NO,DATE <<——> X-CODE,QTYWD

14. Stock of Items

(44) WH-NAME <——> WH-MGR,BUILD-NO

(45) SEC-NO <<——> SH-NAME

(46) ITEM-NO <<——> ITEM-NAME,QTY-ON-HAND

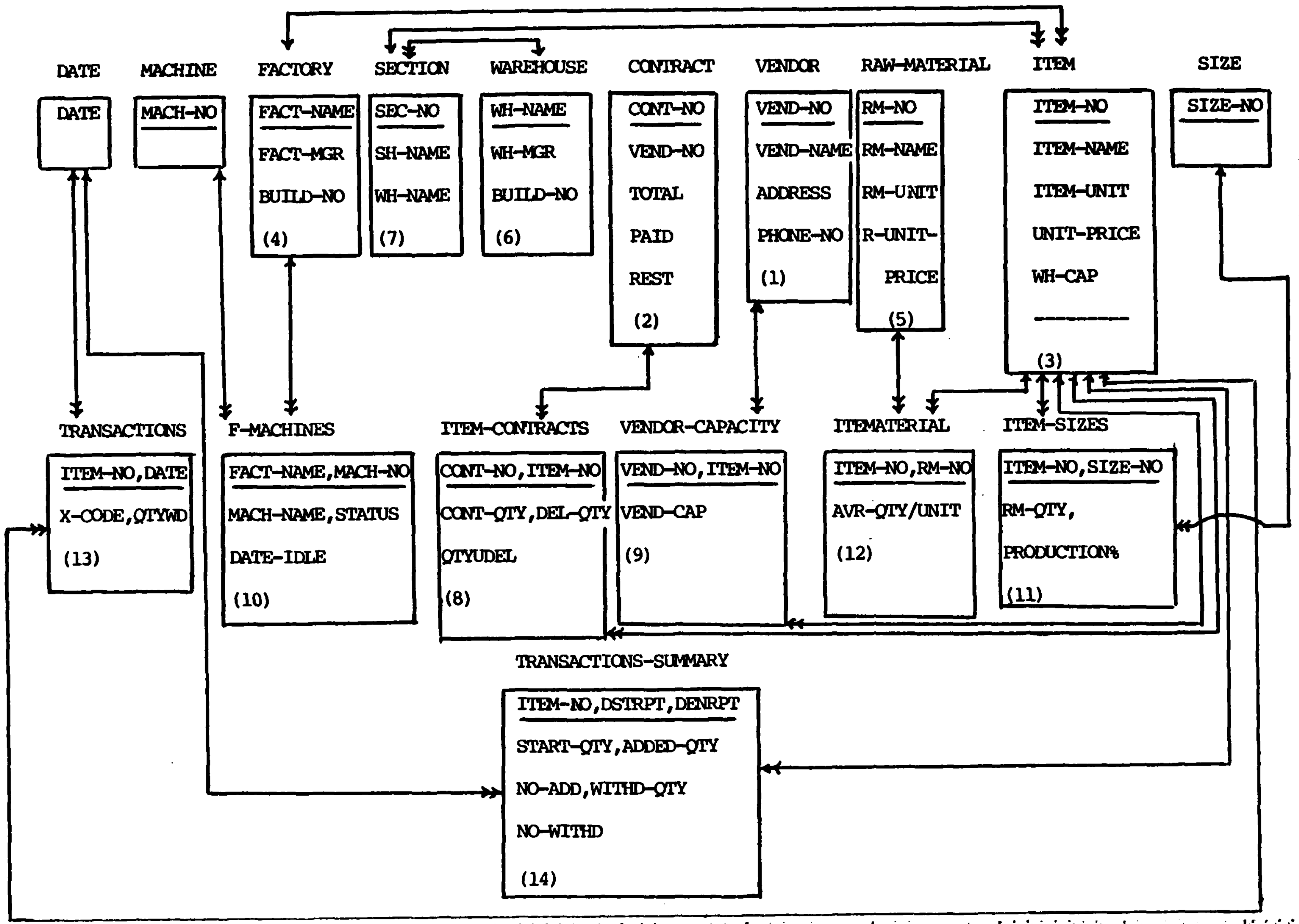
Appendix E

List of Third Normal Form Relations in the Conceptual Model

	Relation Name	Relation Structure
1.	VENDOR	VEND-NO <---> VEND-NAME,ADDRESS,PHONE-NO
2.	CONTRACT	CONT-NO <<---> VEND-NO,TOTAL ,PAID ,REST
3.	ITEM	ITEM-NO <<---> ITEM-NAME,ITEM-UNIT,UNIT-PRICE, WH-CAP,CRL-QTY,QTY-ON-HAND, T-QTYUDEL,DUR-TIME,DESC, SPECS,FACT-NAME,DLW
4.	FACTORY	FACT-NAME <---> FACT-MGR,BUILD-NO
5.	RAW-MATERIAL	RM-NO <<---> RM-NAME,RM-UNIT,RUNIT-PRICE
6.	WAREHOUSE	WH-NAME <---> WH-MGR,BUILD-NO
7.	SECTION	SEC-NO <<---> SH-NAME,WH-NAME
8.	ITEM-CONTRACTS	CONT-NO,ITEM-NO <<---> CONT-QTY,DEL-QTY, QTYUDEL
9.	VENDOR-CAPACITY	VEND-NO,ITEM-NO <<---> VEND-CAP
10.	F-MACHINES	FACT-NAME,MACH-NO <<---> MACH-NAME,STATUS, IDLE-DATE
11.	ITEM-SIZES	ITEM-NO,SIZE-NO <---> RM-QTY,PRODUCTION %
12.	ITEMATERIAL	ITEM-NO,RM-NO <<---> AVR-QTY/UNIT
13.	TRANSACTIONS	ITEM-NO,DATE <<---> X-CODE,QTYWD
14.	TRANSACTIONS-SUMMARY	ITEM-NO,DSTRPT,DENRPT <<---> STRT-QTY, ADDED-QTY, NO-ADD, WITHD-QTY, NO-WITHD

Appendix F

The Conceptual Model



Appendix C

Subset of the
Logical Data Base

Relations From Level 1
ITEM:

ITEM-NO	ITEM-NAME	ITEM-UNIT	UNIT-PRICE	WH-CAP	CRL-QTY	QTY-ON-HAND	T-QTYUDEL	DUR-TIME	DESC (Decsciprion)	SPECS (Specifications)	FACT-NAME	DLW
110C1	Battle dress	one	40.20	500,000	20,000	120,000	600,000	3			Clothes	01-07-84
115C1	Blue cap	one	8.10	1,000,000	10,000	400,000	250,000	3			Clothes	28-06-84
155C1	Gray stocking	pair	1.25	2,000,000	50,000	750,000	750,000	1				01-07-84
150C2	Long sleeve shirt	one	6.50	1,000,000	50,000	400,000	800,000	1			Clothes	29-06-84
080C2	Black sandels	pair	8.00	500,000	20,000	220,000	350,000	1			Shoes	15-10-83

FACTORY:

FACT-NAME	FACT-MGR	BUILD-NO
Clothes	James White	224
Shoes	John Longman	166
Furniture	Robert Ramsey	611
Metals	Jeffrey Coleman	233

RAW MATERIAL:

RM-NO	RM-NAME	RM-UNIT	RUNIT-PRICE
235R1	Serg Wool	Meter	9.50
270R2	Cotton drell	Meter	3.10
299R2	White cotton cloth	Meter	2.20
150R3	Polyester cloth	Meter	4.60
161R4	Black leather	Square foot	4.30
172R4	Brown leather	Square foot	4.70

Relations from Level 2

F-MACHINES:

FACT-NAME	MACH-NO	MACH-NAME	STATUS	IDLE-DATE
Clothes	120	Sewing machine	W	
Clothes	121	Sewing machine	W	
Furniture	204	Electric saw	W	
Furniture	260	Electric saw	I	10-07-84
Furniture	150	Electric drill	W	

ITEM-SIZES

ITEM-NO	SIZE-NO	RM-QTY	PRODUCTION %
110C1	1	3.80	20
110C1	2	3.56	55
110C1	3	3.22	25
150C2	1	2.90	2
150C2	2	2.81	12
150C2	3	2.77	18
150C2	4	2.64	25
150C2	5	2.61	31
150C2	6	2.56	12

ITEMATERIAL:

ITEM-NO	RM-NO	AVR-QTY/UNIT
110C1	235R1	3.523
115C1	290R1	.25
150C2	270R2	2.6701
080C3	161R4	1.06

Appendix H

The Physical Model

```
relation name      relation owner

attribute          aeldeihi
relation           aeldeihi
item               aeldeihi
item_sizes         aeldeihi
indexes            aeldeihi
integrities        aeldeihi
factory            aeldeihi
f_machines         aeldeihi
protect            aeldeihi
itematerial        aeldeihi
tree               aeldeihi
raw_material       aeldeihi


Relation:                item
Owner:                   aeldeihi
Tuple width:             338
Saved until:             Tue Dec 18 00:00:00 1984
Number of tuples:        5
Storage structure:       ISAM file
Relation type:           user relation

attribute name     type   length   keyno.
item_no             c       5        1
item_name           c      30
item_unit           c       6
unit_price          f       8
wh_cap              i       4
crl_qty             i       4
qty_on_hand         i       4
t_qtyudel           i       4
dur_time            i       4
desc                c     120
specs               c     120
fact_name           c      30
dlw                 c      10


Relation:                factory
Owner:                   aeldeihi
Tuple width:             56
Saved until:             Tue Dec 18 00:00:00 1984
Number of tuples:        4
Storage structure:       RANDOM HASH
Relation type:           user relation

attribute name     type   length   keyno.
fact_name           c      25        1
fact_mgr            c      30
build_no            i       4
```

```
Relation:                  raw_material
Owner:                     aeldeihi
Tuple width:               45
Saved until:               Tue Dec 18 00:00:00 1984
Number of tuples:          6
Storage structure:         RANDOM HASH
Relation type:             user relation

attribute name     type  length  Keyno.
rm_no               c       5      1
rm_name             c      30
rm_unit             c      10
runit_price         f       8


Relation:                  f_machines
Owner:                     aeldeihi
Tuple width:               70
Saved until:               Tue Dec 18 00:00:00 1984
Number of tuples:          5
Storage structure:         ISAM file
Relation type:             user relation

attribute name     type  length  Keyno.
fact_name           c      30      1
mach_no             i       4      2
mach_name           c      30
status              c       6
idle_date           c      10


Relation:                  item_sizes
Owner:                     aeldeihi
Tuple width:               11
Saved until:               Tue Dec 18 00:00:00 1984
Number of tuples:          9
Storage structure:         RANDOM HASH
Relation type:             user relation

attribute name     type  length  Keyno.
item_no             c       5      1
size_no             i       4      2
rm_qty              f       8
prod_per            i       4


Relation:                  itematerial
Owner:                     aeldeihi
Tuple width:               14
Saved until:               Tue Dec 18 00:00:00 1984
Number of tuples:          4
Storage structure:         RANDOM HASH
Relation type:             user relation

attribute name     type  length  Keyno.
item_no             c       5      1
rm_no               c       5      2
avr_qty_u           f       8
```

Appendix I

Sample Queries

```
  range of i is item
* retrieve (i.item_name)
* where i.item_no = "150C2"
* range of r is raw_material
* range of m is itematerial
* retrieve (r.rm_no,r.rm_name,r.runit_price)
* where r.rm_no = m.rm_no
* and m.item_no = "150C2"
* range of s is item_sizes
* retrieve (s.size_no,s.rm_qty,s.prod_per)
* where s.item_no = "150C2"
* \g
Executing . . .


|item_name                         |
|----------------------------------|
|Long Sleeve Shirt                 |
|----------------------------------|

|rm_no |rm_name                          |runit_pric|
|-------------------------------------------------|
|27082 |Cotton Drell                     |     3.100|
|-------------------------------------------------|

|size_no      |rm_qty     |prod_per      |
|----------------------------------------|
|            1|      2.900|             2|
|            2|      2.810|            12|
|            3|      2.770|            18|
|            4|      2.640|            25|
|            5|      2.610|            31|
|            6|      2.560|            12|
|----------------------------------------|
(6 tuples)

continue
*
```

Appendix J

The Cutting Plan Program

```
gram cuttingplan (input,output);


***************************************************************************)
                                                                          *)
   DATE              : 2_9_84                                             *)
   VERSION           : 3.0                                                *)
   TITLE             : Cuttingplan                                        *)
   FILENAME          : ali.p                                              *)
   OWNER             : El Deihi                                           *)
   SOFTWARE SYSTEM   : UNIX                                               *)
   OPERATING SYSTEM  : UNIX                                               *)
   LANGUAGE          : PASCAL                                             *)
   USE               : Solving cutting problems by the column             *)
                       generation technique and preparing a               *)
                       cutting plan .                                     *)
   CONTENTS          : Procedures inputdata , printinput ,                *)
                                  printiteration , cibfs ,                *)
                                  pivot , effeciency , comb2 ,            *)
                                  comb3 , comb4 , finalresults ,          *)
                                  printresults .                          *)
   FUNCTION          :                                                    *)
                                                                          *)
***************************************************************************)
```

```
edure effeciency(var no,ix  : list ;
                 var c : rlist ;
                 var iz : sqlist ;
                 var a   : rsqlist ;
                 var z,znew   : real ;
                 var n,it,K,p   : integer ) ;

 i,j   : integer;
 h          : real;

in
 0.0;
 i := 1 to n do
in
for j := 1 to n do
 := h + ix[i] * a[j,i];
;
(h > 1)   then
in
 K := K + 1;
 pivot(no,ix,c,iz,a,znew,n,K);
 it := it + 1;
 z := znew;
 if (p = 1) then
 printiteration(c,iz,z,it,n);
; (* if *)
; (* procedure *)
```

```
(*****************************************************************************)
*                                                                           *)
*     DATE                          : 2_9_84                                *)
*     VERSION                       : 3.0                                   *)
*     NAME                          : effeciency                            *)
*     FUNCTION                      : checks new patterns , ond if          *)
*                                     found that it will imorove the        *)
*                                     solution then procedure pivot is      *)
*                                     called , otherwise the pattern        *)
*                                     is discorded .                        *)
*     INPUTS                        : no , ix , c , iz , a , z , znew ,     *)
*                                     n , it , K , p .                      *)
*     OUTPUTS                       : it , z .                              *)
*     GLOBAL VARIABLES USED         : z , znew , n , it , K , p .           *)
*     GLOBAL VARIABLES CHANGED      : it , z .                              *)
*     GLOBAL TABLES USED            : no , ix , c , iz , a .                *)
*     GLOBAL TABLES CHANGED         : none                                  *)
*     FILES READ                    : none                                  *)
*     FILES WRITTEN                 : none                                  *)
*     MODULES CALLED                : pivot , printiteration                *)
*     CALLING MODULES               : comb2 , comb3 , comb4 .               *)
*                                                                           *)
*     AUTHOR                        : EL Deihi                              *)
*     HISTORY                       :                                       *)
*****************************************************************************)
```

```
rocedure pivot (var no,ix : list ;
                  var c : rlist ;
                  var ix : sqlist ;
                  var a : rsqlist ;
                  var znew : real ;
                  var n,K : integer);

or   i,j,ip    : integer;
     mini,piv,sum : real;
     vol : rlist;

egin

new := 0.0;
ini := 99999.0;
or i := 1 to n do
    val[i] := 0.0;
or i := 1 to n do
egin
   for j := 1 to n do
      val[i] := val[i] + ix[j] * a[i,j];
   if(vol[i] > 0.0)   then
   begin
      sum := c[i] / vol[i];
      if(sum < mini)   then
      begin
         mini := sum;
         ip := i
       end;  (* if *)
    end;  (* if *)
nd; (* for *)
o[ip] := K;
For i := 1 to n do
    ix[i,ip] := ix[i];
iv := vol[ip];
For i := 1 to n do
egin
   if(i <> ip)   then
   begin
      c[i] := c[i] - c[ip] * vol[i] / piv;
      for j := 1 to n do
      a[i,j] := a[i,j] - a[ip,j] * val[i] / piv;
   end; (* if *)
nd; (* for *)
For i := 1 to n do
    a[ip,i] := a[ip,i] / piv;
[ip] := c[ip] /piv;
For i := 1 to n do
    znew := znew + c[i];
nd;   (* procedure *)
```

```
(*************************************************************************)
(*                                                                      *)
(*     DATE                         : 2_9_84                            *)
(*     VERSION                      : 3.0                               *)
(*     NAME                         : pivot                             *)
(*     FUNCTION                     : performes the pivot operation     *)
(*     INPUTS                       : no , ix , c , iz , a , znew , n ,*)
(*                                    k .                               *)
(*     OUTPUTS                      : no , c , iz , a , znew ,          *)
(*     GLOBAL VARIABLES USED        : znew , n , k ,                    *)
(*     GLOBAL VARIABLES CHANGED     : znew                              *)
(*     GLOBAL TABLES USED           : no , ix , c , iz , a .            *)
(*     GLOBAL TABLES CHANGED        : no , c , iz , a .                 *)
(*     FILES READ                   : none                              *)
(*     FILES WRITTEN                : none                              *)
(*     MODULES CALLED               : none                              *)
(*     CALLING MODULES              : effeciency                        *)
(*                                                                      *)
(*     AUTHOR                       : EL Deihi                          *)
(*     HISTORY                      :                                   *)
(*************************************************************************)
```

```
procedure cibfs(var no, noc,lop,nop : list ;
                var c : rlist ;
                var iz : sqlist ;
                var a : rsqlist ;
                var purl,z,mpw,lim,initloss : real ;
                var K,n,lor,rqu,p : integer );

var  zin : real;
     i         : integer;

begin

it := 0;
lim := 0.0;
K := n;
for i := 1 to n do
begin
   no[i] := i;
   noc[i] := trunc(lor / lop[i]);
   purl := purl + lop[i] * nop[i] / lor;
   iz[i,i] := noc[i];
   a[i,i] := 1 / noc[i];
   c[i] := nop[i] /noc[i];
   zin := zin + c[i]
end; (* for *)

z := zin;
initloss := (zin - purl) * 100 / purl;
lim := purl + rqu * mpw / lor;
it := it + 1;
if (p = 1) then
printiteration(c,iz,z,it,n);
end;    (* procedure *)
```

```
/*****************************************************************************
 *
 *      DATE                              :  2_9_9[illegible]
 *      VERSION                           :  3.0
 *      NAME                              :  cihf()
 *      FUNCTION                          :  creates the initial [illegible]
 *                                           possible solutions.
 *
 *      INPUTS                            :  [illegible]
 *      OUTPUTS                           :  no, no, [illegible]
 *                                        :  initbloss, [illegible]
 *
 *      GLOBAL VARIABLES USED             :  new, [illegible]
 *                                           in bloss, [illegible]
 *
 *      GLOBAL VARIABLES CHANGED          :  [illegible]
 *      GLOBAL TABLES USED                :  [illegible]
 *      GLOBAL TABLES CHANGED             :  [illegible]
 *      FILES READ                        :  none
 *      FILES WRITTEN                     :  none
 *      MODULES CALLED                    :  [illegible]
 *      CALLING MODULES                   :  main
 *
 *      AUTHOR                            :  EL Deinz
 *      HISTORY                           :
 *****************************************************************************/
```

```
procedure printiteration (var c : rlist ;
                          var iz : sqlist ;
                          var z : real ;
                          var it,n : integer);

var  i,j,l,middle      : integer;

begin

l := l + 1;
writeln('Table':25,it:4);
writeln('==========':30);
writeln;
write('  Size No  ');
middle := n * 3 ;
for i := 1 to middle do
   write(' ');
write('Patterns');
writeln;
write('             ');
for i := 1 to n do
    write(i:6);
writeln;
write('___________');
for i := 1 to n do
   write('______');
writeln;
for i := 1 to n do
begin
   write(i:6,' ':5);
   for j := 1 to n do
      write(iz[i,j]:6);
   writeln
end; (* for *)
write('___________');
for i := 1 to n do
   write('______');
writeln;
write('No of Rolls  ');
for i := 1 to n do
    write(c[i]:6:2);
writeln;
writeln('Z =   ',z:8:2)
end; (* procedure *)
```

```
(****************************************************************************)
(*                                                                          *)
(*    DATE                         : 2_9_84                                 *)
(*    VERSION                      : 3.0                                    *)
(*    NAME                         : printiteration                         *)
(*    FUNCTION                     : prints intermediate solution           *)
(*                                   steps .                                *)
(*    INPUTS                       : no , c , iz , z , it , n               *)
(*    OUTPUTS                      : none                                   *)
(*    GLOBAL VARIABLES USED        : z , it , n                             *)
(*    GLOBAL VARIABLES CHANGED     : none                                   *)
(*    GLOBAL TABLES USED           : no , c , iz                            *)
(*    GLOBAL TABLES CHANGED        : none                                   *)
(*    FILES READ                   : none                                   *)
(*    FILES WRITTEN                : none                                   *)
(*    MODULES CALLED               : none                                   *)
(*    CALLING MODULES              : cibfs , effeciency                     *)
(*                                                                          *)
(*    AUTHOR                       : EL Deihi                               *)
(*    HISTORY                      :                                        *)
(****************************************************************************)
```

```
procedure printinput (var no,lop,nop : list ;
                      var itemnumb,rawnumb : numb ;
                      var itemname,rawname : name ;
                      var price,mpw : real ;
                      var n,lor : integer);

var     i          : integer;

begin

writeln('INPUT DATA':28);
writeln('==========':28);
writeln;

write('Item No           : ');
for i := 1 to 5 do
    write(itemnumb[i]);
write(' ':15,'R_Material No   : ');
for i := 1 to 5 do
    write(rawnumb[i]);
writeln;

write('Item Name        : ');
for i := 1 to 20 do
   write(itemname[i]);
write('R_Material Name: ');
for i := 1 to 20 do
   write(rawname[i]);
writeln;

write(' ':37,'Unit_Price       : ',price:6:2);
writeln;

write('Requ Quantity   : ',rqu:6,' ':14,'Length of Roll : ',lor:6);
writeln;

write('Number of Sizes: ',n:6);
writeln;

write('max Waste/Unit : ',mpw:6:2);
writeln;

writeln('______________________________________________________________');
writeln(' Size No     Required Length        Number of Pieces');
writeln('______________________________________________________________');
for i := 1 to n do
begin
   write(no[i]:5,' ':12,lop[i]:5,' ':17,nop[i]:5);
   writeln
end;
writeln('______________________________________________________________')

end; (* procedure *)
```

```
(******************************************************************)
(*                                                                *)
(*    DATE                         : 2_9_84                       *)
(*    VERSION                      : 3.0                          *)
(*    NAME                         : printinput                   *)
(*    FUNCTION                     : prints the input data        *)
(*    INPUTS                       : no , lop , nop , itemnumb ,  *)
(*                                   rawnumb , itemname , rawname , *)
(*                                   price , mpw , n , lor .      *)
(*    OUTPUTS                      : none                         *)
(*    GLOBAL VARIABLES USED        : price , mpw , n , lor.       *)
(*    GLOBAL VARIABLES CHANGED     : none                         *)
(*    GLOBAL TABLES USED           : no , lop , nop , itemnumb ,  *)
(*                                   rawnumb , itemname , rawname . *)
(*    GLOBAL TABLES CHANGED        : none                         *)
(*    FILES READ                   : none                         *)
(*    FILES WRITTEN                : none                         *)
(*    MODULES CALLED               : none                         *)
(*    CALLING MODULES              : main                         *)
(*                                                                *)
(*    AUTHOR                       : EL Dejhi                     *)
(*    HISTORY                      :                              *)
(******************************************************************)
```

```
reset(input);
writeln ('Enter the raw material number ');
i := 1;
while not eoln do
begin
    read  (rawnumb[i]);
    i := i + 1
end;  (* while *)
reset(input);
for i := 1 to 30 do
    rawname[i] := ' ';
writeln ('Enter the raw material name up to 30 characters');
i := 1;
while not eoln do
begin
    read  (rawname[i]);
    i := i + 1
end;  (* while *)
writeln ('Enter the price of one unit of raw material in pound');
read  (price);
writeln ('Enter the length of roll in Cm.');
read  (lor);
writeln ('Do U want to print all solution iterations (Y/N = 1/0)');
read (p);
end;
```

```
procedure inputdata (var no,lop,per,nop : list ;
                     var nsc : comb ;
                     var itemnumb,rawnumb : numb ;
                     var itemname,rawname : name ;
                     var price,mpw : real ;
                     var lor,n,rqu : integer);

 var   nc       : integer;
       i        : integer;

begin

reset(input);
writeln ('Enter the item number ');
i := 1;
while not eoln do
begin
   read  (itemnumb[i]);
   i := i + 1
end; (* while *)
reset(input);
for i := 1 to 30 do
  itemname[i] := ' ';
writeln ('Enter the item name up to 30 character');
i := 1;
while not eoln do
begin
   read  (itemname[i]);
   i := i + 1
end; (* while *)
writeln ('Enter the required quantity');
read  (rqu);
writeln ('Enter the number of sizes');
read  (n);
for i := 1 to n do
begin
   writeln ('Enter the length of the size pattern in Cm.');
   read (lop[i]);
   writeln ('Enter the production %');
   read (per[i]);
   nop[i] := trunc(rqu * per[i] / 100);
   no[i] := i
end;
writeln ('Enter the number of different combinations accepted 1,2,or3 ');
read (nc);
for i := 1 to nc do
begin
   writeln ('Enter the accepted combination 2,3,or4');
   read (nsc[i])
end;
writeln ('Enter the maximum permissible waste per unit in Cm.');
read  (mpw);
```

```
(*************************************************************************)
(*                                                                       *)
(*    DATE                         : 2_9_84                              *)
(*    VERSION                      : 3.0                                 *)
(*    NAME                         : inputdata                           *)
(*    FUNCTION                     : Gets the input data from the        *)
(*                                   terminal .                          *)
(*    INPUTS                       : none                                *)
(*    OUTPUTS                      : no , lop , per , nop , nsc ,        *)
(*                                   itemnumb , rownumb , itemname ,     *)
(*                                   rowname , price , mpw , lor , n ,*)
(*                                   rqu .                               *)
(*    GLOBAL VARIABLES USED        : price , mpw , lor , n , rqu .       *)
(*    GLOBAL VARIABLES CHANGED : all                                     *)
(*    GLOBAL TABLES USED           : no , lop , per , nop , nsc ,        *)
(*                                   itemnumb , rownumb , itemname ,     *)
(*                                   rowname .                           *)
(*    GLOBAL TABLES CHANGED        : all                                 *)
(*    FILES READ                   : none                                *)
(*    FILES WRITTEN                : none                                *)
(*    MODULES CALLED               : none                                *)
(*    CALLING MODULES              : main                                *)
(*                                                                       *)
(*    AUTHOR                       : EL Deihi                            *)
(*    HISTORY                      :                                     *)
(*************************************************************************)
```

```
type    list     = array[1..6] of integer;
        sqlist   = array[1..6,1..6] of integer;
        rlist    = array[1..6] of real;
        rsqlist  = array[1..6,1..6] of real;

        comb     = array[1..3] of integer;
        numb     = array[1..5] of char;
        name     = array[1..30] of char;

var     no       : list;        (* size number *)
        lop      : list;        (* length of pattern *)
        per      : list;        (* production % *)
        nop      : list;        (* number of pieces *)
        noc      : list;        (* number of cuts *)
        ix       : list;        (* array used to store created patterns *)
        ic       : list;        (* array used to store integer number of*)
                                (* rolls cut in the optimal solution    *)
        trim     : list;        (* array used to store trimloss in patternes *)
        irem     : list;        (* array used to store fractions of rolls *)
                                (* cut in the optimal solution            *)

        c        : rlist;       (* array used to store number of rolls cut *)
                                (* in the current solution                 *)

        iz       : sqlist;      (* array of all patterns in the current solution

        a        : rsqlist;     (* array of the typical pattern *)

        nsc      : comb;        (* number of sizes combinations *)

        itemnumb : numb;        (* the item number *)
        rawnumb  : numb;        (* the raw material number *)

        itemname : name;        (* the item name *)
        rawname  : name;        (* the raw material name *)


        price    : real;       (* price of one unit of raw material *)
        mpw      : real;       (* maximum permissible waste per unit of item *)
        purl     : real;       (* pure length i.e total length required *)
                               (* with no waste                         *)
        z        : real;       (* number of rolls cut in the old solution *)
        znew     : real;       (* number of rolls cut in the current solution *)
        fsum     : real;       (* integer number of rolls in optimal solution *
        totprice : real;       (* total raw material price *)
        lim      : real;       (* number of rolls equivalent to purl *)
        initloss : real;       (* trimloss in the initial solution *)
        finaloss : real;       (* trimloss in the final solution *)

        rqu      : integer;    (* required quantity *)
        k        : integer;    (* counter of new effecient patterns *)
        n        : integer;    (* number of different sizes required *)
        lor      : integer;    (* length of roll *)
        it       : integer;    (* iteration number *)
        p        : integer;    (* flag to print iterations *)
```

```
(*********************************************************************)
(*                                                                  *)
(*    DATE                       : 2_9_84                           *)
(*    VERSION                    : 3.0                              *)
(*    NAME                       : comb2                            *)
(*    FUNCTION                   : creates new 2 sizes patterns     *)
(*    INPUTS                     : no , ix , noc , lop , c, iz , o ,*)
(*                                 z , znew , n , lor , K , it , p  *)
(*    OUTPUTS                    : ix                               *)
(*    GLOBAL VARIABLES USED      : z , znew , n , lor , K , it , p  *)
(*    GLOBAL VARIABLES CHANGED   : none                             *)
(*    GLOBAL TABLES USED         : no , ix , noc , lop , c , iz , a *)
(*    GLOBAL TABLES CHANGED      : ix                               *)
(*    FILES READ                 : none                             *)
(*    FILES WRITTEN              : none                             *)
(*    MODULES CALLED             : eefeciency                       *)
(*    CALLING MODULES            : main                             *)
(*                                                                  *)
(*    AUTHOR                     : EL Deihi                         *)
(*    HISTORY                    :                                  *)
(*********************************************************************)
```

```
procedure comb2 (var no,ix,noc,lop : list ;
                 var c : rlist ;
                 var iz : sqlist ;
                 var a : rsqlist ;
                 var z,znew : real ;
                 var n,lor,K,it,p : integer);

var  ii,jj,i5,i6,ix2,isum,i: integer;

begin
  for ii := 1 to n-1 do
  begin
    for jj := ii + 1 to n do
    begin
      for i := 1 to n do
        ix[i] := 0;
      i6 := 1;
      while (i6 <= noc[jj] - 1) do
      begin
        ix[jj] := i6;
        ix2 := 0;
        i5 := 1;
        while (i5 <= noc[ii]) and (ix2 <= noc[jj] - 1) do
        begin
          ix[ii] := i5;
          ix2 := i5 + i6;
          isum := i5 * lop[ii] + i6 * lop[jj];
          if (ix2 >= noc[ii]) and (ix2 <= noc[jj]) and (isum <= lor)  then
            effeciency(no,ix,c,iz,a,z,znew,n,it,K,p);
            i5 := i5 + 1;
        end; (* while  *)
        i6 := i6 + 1;
      end; (* while *)
    end; (* for *)
  end; (* for *)
end;    (* procedure *)
```

```
(*****************************************************************************)
(*                                                                          *)
(*    DATE                        : 2_9_84                                  *)
(*    VERSION                     : 3.0                                     *)
(*    NAME                        : comb3                                   *)
(*    FUNCTION                    : creates new 3 sizes patterns            *)
(*    INPUTS                      : no , ix , noc , lop , c, iz , o ,*)
(*                                  z , znew , n , lor , k , it , p         *)
(*    OUTPUTS                     : ix                                      *)
(*    GLOBAL VARIABLES USED       : z , znew , n , lor , k , it , p         *)
(*    GLOBAL VARIABLES CHANGED    : none                                    *)
(*    GLOBAL TABLES USED          : no , ix , noc , lop , c , iz , o *)
(*    GLOBAL TABLES CHANGED       : ix                                      *)
(*    FILES READ                  : none                                    *)
(*    FILES WRITTEN               : none                                    *)
(*    MODULES CALLED              : eefeciency                              *)
(*    CALLING MODULES             : main                                    *)
(*                                                                          *)
(*    AUTHOR                      : EL Deihi                                *)
(*    HISTORY                     :                                         *)
(*****************************************************************************)
```

```
procedure comb3 (var no,ix,noc,lop : list ;
                 var c : rlist ;
                 var iz : sqlist ;
                 var a : rsqlist ;
                 var z,znew : real ;
                 var n,lor,K,it,p : integer);

var  ii,jj,KK,i4,i5,i6,ix2,ix3,isum,i: integer;

begin
  for ii := 1 to n-2 do
  begin
    for jj := ii + 1 to n -1 do
    begin
      for KK := jj + 1 to n do
      begin
        for i := 1 to n do
          ix[i] := 0;
        i6 := 1;
        while (i6 <= noc[KK] - 2) do
        begin
          ix[KK] := i6;
          i5 := 1;
          ix2 := 0;
          while(i5 <= noc[jj] - 1) and (ix2 <= noc[KK] - 2) do
          begin
            ix[jj] := i5;
            ix2 := i6 + i5;
            i4 := 1;
            ix3 := 0;
            while (i4 <= noc[ii]) and (ix3 <= noc[jj] - 1) do
            begin
              ix[ii] := i4;
              ix3 := i4 + i5 + i6;
              isum := i4 * lop[ii] + i5 * lop[jj] + i6 * lop[KK];
              if (ix3 >= noc[ii]) and (ix3 <= noc[KK]) and (isum <= lor) then
            effeciency(no,ix,c,iz,a,z,znew,n,it,K,p);
            i4 := i4 + 1;
            end; (* while  *)
            i5 := i5 + 1;
          end; (* while *)
          i6 := i6 + 1;
        end; (* while *)
      end; (* for *)
    end; (* for *)
  end; (* for *)
end;    (* procedure *)
```

```
(*****************************************************************)
(*                                                               *)
(*    DATE                       : 2_9_84                        *)
(*    VERSION                    : 3.0                           *)
(*    NAME                       : comb4                         *)
(*    FUNCTION                   : creates new 4 sizes patterns  *)
(*    INPUTS                     : no , ix , noc , lop , c, iz , o ,*)
(*                                 z , znew , n , lor , k , it , o  *)
(*    OUTPUTS                    : ix                            *)
(*    GLOBAL VARIABLES USED      : z , znew , n , lor , k , it , o  *)
(*    GLOBAL VARIABLES CHANGED   : none                          *)
(*    GLOBAL TABLES USED         : no , ix , noc , lop , c , iz , o *)
(*    GLOBAL TABLES CHANGED      : ix                            *)
(*    FILES READ                 : none                          *)
(*    FILES WRITTEN              : none                          *)
(*    MODULES CALLED             : eefeciency                    *)
(*    CALLING MODULES            : main                          *)
(*                                                               *)
(*    AUTHOR                     : EL Deibi                      *)
(*    HISTORY                    :                               *)
(*****************************************************************)
```

```
procedure comb4 (var no,ix,noc,lop : list ;
                 var c : rlist ;
                 var iz : sqlist ;
                 var a : rsqlist ;
                 var z,znew : real ;
                 var n,lor,K,it,p : integer);

var  ii,jj,KK,ll,i3,i4,i5,i6,ix2,ix3,ix4,isum,i: integer;

begin
  for ii := 1 to n-3 do
  begin
    for jj := ii + 1 to n - 2  do
    begin
      for KK := jj + 1 to n - 1 do
      begin
        for ll := KK + 1 to n do
        begin
          for i := 1 to n do
            ix[i] := 0;
            i6 := 1 ;
            while (i6 <= noc[ll] - 3) do
            begin
               ix[ll] := i6;
               i5 := 1;
               ix2 := 0;
               while (i5 <= noc[KK] - 2) and (ix2 <= noc[ll] - 3) do
               begin
                 ix[KK] := i5;
                 ix2 := i6 + i5;
                 i4 := 1;
                 ix3 := 0;
                 while (i4 <= noc[jj] - 1) and (ix3 <= noc[KK] - 2) do
                 begin
                   ix[jj] := i4;
                   ix3 := i6 + i5 + i4;
                   i3 := 1;
                   ix4 := 0;
                   while( i3 <= noc[ii]) and (ix4 <= noc[jj] - 1) do
                   begin
                     ix[ii] := i3;
                     ix4 := i3 + i4 + i5 + i6;
                     isum := i3 * lop[ii] + i4 * lop[jj] + i5 * lop[KK] + i6
                     if (ix4 >= noc[ii]) and (ix4 <= noc[ll]) and (isum <= lo
                     then
                     effeciency(no,ix,c,iz,a,z,znew,n,it,K,p);
                     i3 := i3 + 1;
                    end; (* while *)
                    i4 := i4 + 1;
                 end; (* while *)
                 i5 := i5 + 1;
               end; (* while  *)
               i6 := i6 + 1;
            end; (* while *)
        end; (* for *)
      end; (* for *)
    end; (* for *)
  end; (* for *)
end;    (* procedure *)
```

```
(*************************************************************************)
(*                                                                      *)
(*    DATE                        : 2_9_84                              *)
(*    VERSION                     : 3.0                                 *)
(*    NAME                        : final result                        *)
(*    FUNCTION                    : computes the information required*)
(*                                  to print the cutting plan .         *)
(*    INPUTS                      : ic , ix , lop , nop , trim , irem*)
(*                                  c , iz , n , lor , totprice ,       *)
(*                                  price , fsum , purl , finaloss .    *)
(*    OUTPUTS                     : ic  , ix trim , irem , c , iz ,     *)
(*                                  totprice .                          *)
(*    GLOBAL VARIABLES USED       : n , lor , totprice , price ,        *)
(*                                  fsum , purl , finaloss ,            *)
(*    GLOBAL VARIABLES CHANGED    : totprice , fsum , finaloss .        *)
(*    GLOBAL TABLES USED          : ic , ix , lop , nop , trim ,        *)
(*                                  irem , c , iz .                     *)
(*    GLOBAL TABLES CHANGED       : ic , ix , trim , irem , c , iz .    *)
(*    FILES READ                  : none                                *)
(*    FILES WRITTEN               : none                                *)
(*    MODULES CALLED              : none                                *)
(*    CALLING MODULES             : main                                *)
(*                                                                      *)
(*    AUTHOR                      : EL Deihi                            *)
(*    HISTORY                     :                                     *)
(*************************************************************************)
```

```
procedure finalresult (var ic,ix,lop,nop,trim,irem : list ;
                       var c : rlist ;
                       var iz : sqlist ;
                       var n,lor : integer ;
                       var totprice,z,price,fsum,purl,finaloss : real);
var i,j,ul,flag : integer ;

begin
  for i := 1 to n do
  begin
    ix[i] := 0;
    ic[i] := trunc(c[i])
  end; (* for *)
  for i := 1 to n do
  begin
    trim[i] := 0;
    ul := 0;
    for j := 1 to n do
    begin
      ix[i] := ix[i] + ic[j] * iz[i,j];
      ul := ul + lop[j] * iz[j,i]
    end; (* for *)
    trim[i] := lor - ul;
    irem[i] := nop[i] - ix[i];
    fsum := fsum + c[i] - ic[i]
  end; (* for *)
  for i := 1 to n do
  begin
    flag := 1;
    for j := 1 to n do
      if (iz[j,i] > irem[j]) then flag := 0;
    if (flag = 1) then
    begin
      for j := 1 to n do
        irem[j] := irem[j] - iz[j,i];
      ic[i] := ic[i] + 1;
      fsum := fsum - 1;
    end; (* if *)
  end; (* for *)
  totprice := z * lor * price;
  finaloss := (z - purl) * 100 / purl;
end; (* procedure *)
```

```
(**********************************************************************)
(*                                                                    *)
(*     DATE                       : 2_9_84                            *)
(*     VERSION                    : 3.0                               *)
(*     NAME                       : printresult                       *)
(*     FUNCTION                   : prints the optimal cutting pClan  *)
(*     INPUTS                     : lop , nop , irem , ic , trim ,    *)
(*                                  iz , n , totprice , z , fsum ,    *)
(*                                  initloss , finaloss .             *)
(*     OUTPUTS                    : none                              *)
(*     GLOBAL VARIABLES USED      : n , totprice , z , fsum ,         *)
(*                                  initloss , finaloss .             *)
(*     GLOBAL VARIABLES CHANGED   : none                              *)
(*     GLOBAL TABLES USED         : lop , nop , irem , ic , trim ,iz  *)
(*     GLOBAL TABLES CHANGED      : none                              *)
(*     FILES READ                 : none                              *)
(*     FILES WRITTEN              : none                              *)
(*     MODULES CALLED             : none                              *)
(*     CALLING MODULES            : main                              *)
(*                                                                    *)
(*     AUTHOR                     : El Deihi                          *)
(*     HISTORY                    :                                   *)
(**********************************************************************)
```

```
procedure printresult (var lop,nop,irem,ic,trim : list ;
                       var iz : sqlist ;
                       var n : integer ;
                       var totprice,z,fsum,initloss,finaloss : real);

var i,j,l,middle : integer ;

begin
l := n + 1;
writeln('CUTTING PLAN':30);
writeln('============':30);
writeln;
write('______________________________');
for i := 1 to l do
   write('______');
writeln;
write(' SIZE  LENGTH  NO OF PECIES ');
middle := n * 3 ;
for i := 1 to middle do
   write(' ');
write('PATTERNS');
writeln;
write('______________________________');
for i := 1 to l do
   write('______');
writeln;
for i := 1 to n do
begin
  write(i:4,' ':3,lop[i]:4,' ':4,nop[i]:8,' ':5);
  for j := 1 to n do
    write(iz[i,j]:6);
  write(irem[i]:6);
  writeln
end;(* for *)
write('______________________________');
for i := 1 to l do
   write('______');
writeln;
write('  NO OF ROLLS                ');
for i := 1 to n do
  write(ic[i]:6);
write(' ':2,fsum:6:2);
writeln;
write('  LOSS PER ROLL             ');
for i := 1 to n do
  write(trim[i]:6);
writeln;
write('______________________________');
for i := 1 to l do
  write('______');
writeln;
write (' TOTAL No OF ROLLS :',z:8:2);
writeln;
write (' TOTAL PRICE         : ',totprice:8:2);
writeln;
write (' INITIAL LOSS        : ',initloss:6:5,'%');
writeln;
write (' FINAL LOSS          : ',finaloss:6:5,'%');
end; (* procedure *)
```

```
****************************************************************************)

egin   (* MAIN *)

nputdata (no,lop,per,nop,nsc,itemnumb,rownumb,itemname,rowname,price,mpw,
              lor,n,rqu);

rintinput(no,lop,nop,itemnumb,rownumb,itemname,rowname,price,mpw,n,lor);

ibfs(no,noc,lop,nop,c,iz,a,purl,z,mpw,lim,initloss,K,n,lor,rqu,p);

if (nsc[1] = 2) and (z > lim) then
omb2(no,ix,noc,lop,c,iz,a,z,znew,n,lor,K,it,p);
if ((nsc[1] = 3) or (nsc[2] = 3)) and (z > lim) then
omb3(no,ix,noc,lop,c,iz,a,z,znew,n,lor,K,it,p);
if ((nsc[1] = 4) or (nsc[2] = 4) or (nsc[3] = 4)) and (z > lim) then
omb4(no,ix,noc,lop,c,iz,a,z,znew,n,lor,K,it,p);

finalresult(ic,ix,lop,nop,trim,irem,c,iz,n,lor,totprice,z,price,fsum,purl,fin
printresult(lop,nop,irem,ic,trim,iz,n,totprice,z,fsum,initloss,finalloss);

end.


%
```

Appendix K

Sample Problem and Solution

```
                    INPUT DATA
                    ==========

Item No          : 15002                R_Material No  : 27082
Item Name        : Long Sleeve Shirt    R_Material Name: Cotton Drell
                                        Unit_Price     :   3.10
Requ Quantity  :  20000                 Length of Roll :   4000
Number of Sizes:      6
max Waste/Unit :   0.00

-------------------------------------------------------
 Size No     Required Length       Number of Peices

-------------------------------------------------------
    1              290                   400
    2              281                  2400
    3              277                  3600
    4              264                  5000
    5              261                  6200
    6              236                  2400

-------------------------------------------------------



                    CUTTING PLAN
                    ============

--------------------------------------------------------------------------------
 SIZE  LENGTH  NO OF PECIES                       PATTERNS

--------------------------------------------------------------------------------
   1     290        400           0     10      0      0      0      1      8
   2     281       2400           1      2      1      0      4      0      2
   3     277       3600           4      1      5      5      0      4      1
   4     264       5000           0      0      0      5      5      5      5
   5     261       6200          10      1      6      3      4      0      1
   6     236       2400           0      0      3      2      2      3      0

--------------------------------------------------------------------------------
  NO OF ROLLS                   296     27     12    368    509    122    1.19
  LOSS PER ROLL                   1      0      0      0      0      2

--------------------------------------------------------------------------------
 TOTAL No OF ROLLS : 1335.19
 TOTAL PRICE       :  16556296.30
 INITIAL LOSS      :  2.31681%
 FINAL LOSS        :  0.01013%%
```

Vita

Col Aly I. El Deihi was born on 1 April 1945 in El Behera, Egypt. He graded from high school in Shubrakhit, Behera in 1961 and attended Alexandria University, Alexandria, Egypt, from which he received the degree of Bachelor of Science in Chemistry in July 1965. Upon graduation he was employed as an assistant teacher for the Faculty of Science, Alexandria University. He entered the Egyptian Army on active duty in February 1967. He spent three months in military training in the Military Technical College, Cairo, Egypt. Upon receiving his commission in May 1967 he was assigned to the Ordnance Department where he served in different positions in the Ordnance Training Center, the Main Ordnance Depot, and the Logistics Institute.

In 1975 he attended a Computer Programming Course in the Military Technical College, Cairo, Egypt for ten months. In 1977 he attended the same college, from which he received a Diploma in Operations Research in March 1980. Upon graduation he was reassigned to the Operations Research Department. While there he participated in different Operations Research and Computer related activities.

In 1983 he was selected to come to the USA, to attend the Air Force Institute of Technology, School of Engineering, to study for a Masters Degree in Computer Systems.

Permanent Address: Aly I. El Deihi
Egypt, Cairo
Medinet Masr, Youssef Abbas St.
Medinet El Tawfik, Building 9, App. 33

ASSIFIED

CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

RT SECURITY CLASSIFICATION LASSIFIED	1b. RESTRICTIVE MARKINGS
RITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
ASSIFICATION/DOWNGRADING SCHEDULE	
RMING ORGANIZATION REPORT NUMBER(S) T/GCS/MATH/84D-3	5. MONITORING ORGANIZATION REPORT NUMBER(S)

. OF PERFORMING ORGANIZATION ıool of Engineering	6b. OFFICE SYMBOL *(If applicable)* AFIT/ENG	7a NAME OF MONITORING ORGANIZATION
IESS *(City, State and ZIP Code)* : Force Institute of Technology .ght-Patterson AFB, OH 45433		7b. ADDRESS *(City, State and ZIP Code)*
: OF FUNDING/SPONSORING ANIZATION	8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

IESS *(City, State and ZIP Code)*	10. SOURCE OF FUNDING NOS.			
	PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
: *(Include Security Classification)* : Box 19				

ONAL AUTHOR(S)
/ Ismail El Deihi, Colonel, Egyptian Army

E OF REPORT Thesis	13b. TIME COVERED FROM ________ TO ________	14. DATE OF REPORT *(Yr., Mo., Day)*	15. PAGE COUNT 126

LEMENTARY NOTATION [illegible]

COSATI CODES			18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
	GROUP	SUB. GR.	Data Base, ~~Data Base Design~~, Relational Data Base, ~~Data Base Implementation~~, Linear programming, Cutting problem, computer Program, Pascal

RACT *(Continue on reverse if necessary and identify by block number)*

tle: Development of a Consolidated Data Base for Stock Control and Production Management.

esis Chairman: Dr. Henry Potoczny

Approved for public release: IAW AFR 190-17.
[illegible] 11 Feb 85
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB OH 45433

RIBUTION/AVAILABILITY OF ABSTRACT SIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
IE OF RESPONSIBLE INDIVIDUAL r. Henry Potoczny	22b. TELEPHONE NUMBER *(Include Area Code)* 255-3098	22c. OFFICE SYMBOL AFIT/ENC

RM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

→ A relational data base for stock control and production management in one of the main logistics depots in Egypt was developed and partially implemented on INGRES.

Top-down analysis of the depot was carried out, entities were determined, and an entity relationship chart was drawn. The areas of stock control and production management were chosen for applying the first data base project.

Information requirements in the two specified areas were analysed, data elements were listed, and the individual user's views were determined. A set of third normal form relations were derived and used as a basis of the design. The conceptual model was developed using a structured technique that yielded the model in a canonical form.

A subset of the conceptual model was mapped to the logical model in relational form, and the logical data base was implemented on INGRES and tested by running some queries against it.

In conjunction with the data base design, linear programming techniques for solving the cutting problem were studied. An efficient algorithm was selected, modified to minimize time and space requirements, and coded in PASCAL. The system was tested by solving some problems, using data retrieved from the data base.

Originator-supplied keywords included: ⇒ / [illegible]

Finally, several recommendations were suggested in order to make the implmentation of the data base easier and successful.